Pontificia Universidad Católica del Ecuador

Facultad de Ingeniería
Escuela de Sistemas

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS

"Implementación de una aplicación web para administrar Información de requerimientos de Fundaciones y Organizaciones Nacionales e Internacionales que pudieren financiar proyectos del Área de Acción Social de la Facultad de Ingeniería PUCE."

AUTOR
DARSHAN ZAMORA TORRES

DIRECTOR
FABIÁN DE LA CRUZ

QUITO, 2013

Dedicatoria

A mi abuelita Luz María Torres, que aun que ya no esta a mi lado, su deseo fue siempre la culminación de mi carrera.

Agradecimiento

Agradezco a Dios:

Por enseñarme valores, amor y dedicación a través del ejemplo, de mis amados padres.

Que puso extraordinarios hermanos a mi lado, infundiendo ánimo, coraje y perseverancia.

Quien me rodeo de los mejores amigos, compañeros, profesores quienes con paciencia y alegría caminaron junto a mi compartiendo vivencias y experiencias.

Cuya infinita bondad me permitió conocer a personas inolvidables, que aunque ya no están aquí, en su momento dejaron invaluables enseñanzas y recuerdos en mi vida.

Por que en el presente trabajo se ve materializado la esencia de cada maravillosa persona que Dios, en un determinado tiempo y lugar, supo poner en mi camino.

"Todo lo puedo en Cristo que me fortalece."
Filipenses 4:13

" Pues mirad, hermanos, vuestra vocación, que no sois muchos sabios según la carne, ni muchos poderosos, ni muchos nobles;sino que lo necio del mundo escogió Dios, para avergonzar a los sabios; y lo débil del mundo escogió Dios, para avergonzar a lo fuerte."
1 Corintios 1:27

Tabla de Contenidos

Introducción

El Departamento de Acción Social de la PUCE en su deber para con el país, desarrolla múltiples proyectos en pro de la comunidad, que con ayuda de alumnos y profesores de las distintas Facultades de la Universidad, se han puesto en marcha ayudando a mucha gente de nuestro país.

Por la gran demanda que ha existido a través del tiempo, la mayoría de proyectos han sido cumplidos, pero otros han quedado archivados en papel o en medios digitales en espera de presupuesto para hacerlos realidad.

De aquí nace la necesidad de implementar una aplicación web para administrar información de requerimientos de fundaciones y organizaciones nacionales e internacionales, que pudieren financiar proyectos del Área de Acción Social de la Facultad de Ingeniería PUCE, utilizando software libre.

CAPÍTULO 1

En el presente capítulo se da una breve introducción a las herramientas del software que se usarán en el desarrollo de la aplicación, de igual modo se introduce el uso de la metodología: se detalla, define y explica su utilización en proyecto de software. A modo de entender rápidamente que hay detrás de la aplicación.

1.Definición de Herramientas de Software a usar.

1.1.Software

Son las instrucciones lógicas entendidas por un equipo especifico (computador, teléfono móvil, TV, refrigerador, entre otros) para realizar una determinada función, como puede ser software del sistema (Sistema Operativo, controladores, actualizaciones) software de aplicación (procesador de texto, videojuegos, manejo del inventario de una empresa, etc.) o software de programación (compiladores, intérpretes, etc.)

1.2. Software Libre

Se refiere a la libertad que tienen los usuarios del software de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software y no precisamente indica que es gratis.
Es decir que los usuarios tienen acceso a las 4 libertades esenciales que son:

- Libertad 0. Ejecutar el programa para cualquier propósito
- Libertad 1. Estudiar como trabaja el programa y cambiarlo para que haga lo que se desee
- Libertad 2. Redistribuir copias para que ayuden al prójimo
- Libertar 3. Distribuir copias de sus versiones modificadas a terceros

El acceso al código fuente es necesario para las libertades 1 y 3.

1.3. Apache

Es un servidor HTTP que mantiene su código abierto y funciona sobre sistemás operativos basados en Unix y Windows. El principal objetivo de apache es proveer un servidor seguro, eficiente y expansible.
Apache es usado para lanzar tanto páginas web estáticas como dinámicas. Es muy utilizada con PHP y MySQL, haciendo un vínculo totalmente compatible, además puede ejecutar y lanzar aplicaciones en Perl, Python, Ruby, dotNet, y Java si es enlazado con un servidor Tomcat

1.4. HTML

Son las siglas **H**yper **T**ext **M**arkup **L**anguage que quiere decir formato de marcaje de hipertexto. Es el estándar usado para la estructuración de la forma de los objetos en una página web como son: el texto, las imágenes y demás contenido multimedia.
El HTML introduce el formato de la página web por medio de etiquetas (<,>) las cuales envuelven las sentencias que encierran el comportamiento del objeto a presentarse en la página web.

Un elemento generalmente tiene una etiqueta de apertura (<nombre-de-elemento>) y una etiqueta de cierre (</nombre-de-elemento>). Los atributos del elemento están contenidos en la etiqueta de inicio y el contenido está ubicado entre las dos etiquetas (<nombre-de-elemento atributo="valor">Contenido</nombre-de-elemento>). Algunos elementos, tales como <br>, no tienen contenido ni llevan una etiqueta de cierre.

En síntesis, un documento HTML contiene las instrucciones en donde estarán ubicados el texto, títulos y demás elementos en una página web, el navegador de Internet como es Firefox, Internet Explorer, Safari entre otros que traduce esas etiquetas, las presenta en la página web como deseamos que se visualice.

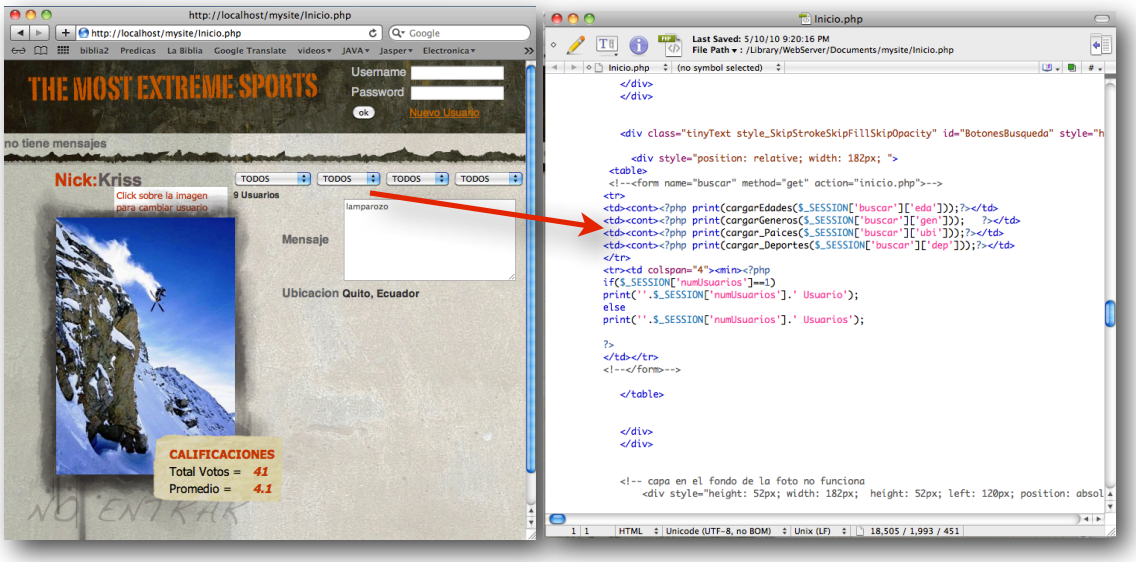


Figura 1: *Página web. En la derecha el código HTML y PHP que generan tal página.*
Elaborado por: Darshan Zamora

1.5. PHP

PHP es un lenguaje de programación de alto nivel orientado a objetos y de código abierto interpretado por el lado del servidor, el cual puede ser embebido dentro de páginas HTML. Gran parte de su sintaxis fue tomada de C, Java y Perl con algunas características propias de PHP.

La meta de PHP es permitir que los creadores de páginas web puedan crear páginas dinámicas de forma rápida y sencilla, aunque en la actualidad se pueden hacer cosas mucho más especializadas. PHP significa PHP Hypertext Preprocessor.

1.6. MySQL

Es un motor gestor de bases de datos relacionales multihilo y multiusuario que posee doble licenciamiento: es libre para proyectos y aplicaciones que se apegan a la licencia GNU GPL; y por otro lado es privativo para proyectos realizados por empresas que así lo deseen.

MySQL ha venido a ser la base de datos de código abierto más popular debido a su alto funcionamiento y su fácil uso. Muchas de las grandes organizaciones de rápido crecimiento como son Facebook, Google, Adobe entre otras usan MySQL en sus aplicaciones. Además Mysql es multiplataforma ya que corre en Linux, Windows, Mac OS, Solaris, IBM entré otras, dotándole de flexibilidad. Posee una gran variedad de herramientas para que esta base de datos trabaje en las mejores condiciones.

1.7. JavaScript

JavaScript es un lenguaje de programación orientado a objetos basados en prototipos que se ejecutan del lado del Cliente, siendo el navegador web el encargado de interpretarlo aunque es posible usarlo en entornos sin navegador web.

La sintaxis es similar a Java y a C++ con el objetivo de reducir la curva de aprendizaje.

1.8. CodeIgniter

CodeIgniter es un poderoso framework construido para desarrollar aplicaciones bajo PHP, que se destaca por su bajo impacto de aprendizaje y la posibilidad de realizar proyectos de manera rápida, ya que posee bibliotecas para tareas comunes como son: validación de datos, envío de emails, accesos a base de datos, manipulación de imágenes, entre otras. De esta manera el desarrollador de la aplicación, se enfoca en el proyecto, minimizando el tiempo de codificación de tareas comunes.

Ademas de implementar la arquitectura MVC (Modelo Vista Controlador)[1] en su estructura, maneja una buena separación entre la lógica y la interfaz gráfica con lo cual se permite que los Diseñadores de la Interface creen plantillas a las cuales será sencillo incrustarle la manipulación de la lógica del negocio.

[1] MVC se detalle en la pagina 82

1.9.Programación Extrema

La programación extrema es una metodología de programación que pertenece a las llamadas metodologías Ágiles para el desarrollo del software, en los últimos años a demostrado mucha aceptación por su rapidez y retroalimentación, sin embargo, ha sido también criticada por quienes usan otras metodología de Ingeniería del software por su falta de documentación y planificación.

La Programación extrema o XP se fundamenta en los siguientes Valores:

- **Simplicidad**, la codificación y el diseño deben ser lo más simples posible, ya que de este modo se agilita el desarrollo y el mantenimiento se simplifica.
- **Comunicación**, a través del código, mientras sea más simple, será más fácil de entender, además debe ser auto documentado. Así, al darle mantenimiento, cualquier programador puede entender que es lo que está haciendo esa parte del código. De igual modo, es importante tener un estándar en la forma de usar los nombres de funciones, clases y variables para no crear confusión.
- **Retroalimentación**, al tener tiempos de vida muy cortos, la interacción con el cliente es continua, de modo que los errores o malentendidos son corregidos en ese instante y al tener pruebas unitarias frecuentes, se puede descubrir fallos en menor tiempo.
- **Coraje,** se ve implícito en diseñar, puesto que implica el diseñar para hoy y no enfocarse en el futuro, pues de un modo pesimista este futuro podría no existir y se perdería todo ese tiempo. Se lo ve también cuando es necesario desechar una parte del código por que está obsoleto sin importar el tiempo invertido en él. Del mismo modo implica persistencia al enfrentar un problema complejo.
- **Respeto,** es evidente entre los programadores, siendo notable hacia el trabajo en su búsqueda de la calidad y excelencia del producto, de modo que se obtiene una mejora entre el equipo, consiguiendo un entorno de trabajo amigable y apto para la producción de un producto cada vez mejor.

1.9.1.Fundación del Desarrollo Ágil de software.

Es un manifiesto y la declaración a la Interdependencia desarrollada por un grupo de expertos del Software que se reunieron en Utah para dar un boceto de lo que conocemos como “Ágile Manifiesto”

> Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:
> **Indivíduos e interacciones** sobre procesos y herramientas.
> **Software funcionando** sobre documentación extensiva.
> **Colaboración con el cliente** sobre negociación contractual.
> **Respuesta ante el cambio** sobre seguir un plan.
> Esto es, aunque valoramos los elementos de la derecha,valoramos más los de la izquierda2.

A más de los 4 valores están los 12 principios.

1. Nuestra prioridad más alta es la satisfacción del cliente a través de la pronta y continua entrega de producto de software funcional.
2. Son bienvenidos los cambios en requerimientos, aún en una etapa avanzada del desarrollo. “Ágile Processes” toma el cambio como ventaja para la competitividad del cliente.
3. Entrega frecuente de Software funcionando desde un par de semanas a un par de meses con preferencia en la más corta escala de tiempo.
4. El desarrollador y el usuario del software deben trabajar juntos a diario a través del desarrollo del proyecto.
5. Construir el proyecto alrededor de motivaciones individuales, dándole un ambiente y un soporte que éste necesite y confiando en que tendrán el trabajo hecho.
6. La más eficiente y efectiva manera de conducir la información hacia y con el equipo de desarrollo del software es en una conversación cara a cara.
7. El software funcionando es la primera medida de progreso.
8. El Proceso Ágil promueve un sustanciable desarrollo. Los auspiciantes, desarrolladores y usuarios deben estar dispuestos a mantener una paz constante e indefinida.
9. La atención continua a técnicas de excelencia y buen diseño aumentan la agilidad
10. Simplicidad. El arte de maximizar la cantidad de trabajo no hecho es esencial.

11. La mejor arquitectura, requerimientos y diseño emergen de la propia organización del equipo.
12. A intervalos regulares, el equipo refleja como llegaron a ser más efectivos, entonces se afinan y ajustan su comportamiento como consecuencia.

Mientras el Ágile Manifiesto trata con el desarrollo de software, el administrador de Proyectos Ágile estableció una declaración de Interdependencia en la que un grupo de expertos reunidos en el año 2005 se centró más en el lado de la gestión de proyectos.

Declaración de Interdependencia[2].

- Nosotros somos una comunidad de líderes de proyectos que son altamente exitosos y entregan resultados. De aquí estos resultados.
- Nosotros aumentamos el retorno en inversión por hacer un continuo flujo de valores de nuestro enfoque
- Nosotros entregamos resultados confiables por enganche de clientes, interacciones frecuentes y compartiendo el dominio.
- Nosotros nos anticipamos a la incertidumbre y la manejamos a través de interacciones y adaptación.
- Nosotros desatamos creatividad e innovación por el reconocimiento que los individuos son la última fuente de valores y creando un medio ambiente donde podemos hacer la diferencia.
- Nuestro mayor rendimiento es a través de un grupo obligado por los resultados y por compartir responsabilidad para un equipo efectivo.
- Nuestra mejor eficacia y confiabilidad a través de estrategias especificas a ciertas situaciones, procesos y prácticas.

De todos modos el “Ágile Manifiesto” y la Declaración de Independencia vino primero de mentes expertas o después de que ellos habían estado de algún modo influenciados por Scrum o por algún otro proceso ágil existente.

[2] Jim Highsmith, Declaration of Interdependence http://pmdoi.org Acceso: (31/07/2012)

1.9.2. Orígenes de SCRUM

Históricamente, el término Scrum vino de un artículo publicado por Hirotaka Takeuchi y Ikujiru Nonaka en la Revista de Negocios de Harvard (Harvard Business Review) en 1986.

En esa publicación titulada “The New New [3] product development game” Takeuchi describe una aproximación holística en la cual el equipo de proyecto está hecho de un pequeño equipo multi-funcional, trabajando exitosamente hacia un objetivo común.

Mientras Jeff Sutherland y Ken Schwaber trabajaban en construir una herramienta de Análisis y Diseño Orientada a objetos (OOAD) para Easel [4], Jeff, en aquel entonces *VP of Ingeniería* de Easel, se dio cuenta de que su equipo de software necesita una versión mejorada de desarrollo rápido de aplicaciones. Él quería que fuera un proceso similar a scrum, donde al final de una corta iteración, el *Director Ejecutivo (CEO) podría ver trabajando al código, demostrando en lugar de los diagramas de Gantt en Papel.*

1.9.3. Historia de Scrum

Si una organización falla en obtener sus logros, esto no siempre es por que los miembros no han trabajado lo suficientemente duro. Más bien, a menudo, una organización falla sus logros porque ha perdido sus ***objetivos*** *y ha perdido el* ***enfoque****.*

El mundo corporativo es un mundo caótico y el número de cosas que pueden distraer al equipo y noquearlo, es por supuesto casi innumerable. El principal propósito de Scrum es ayudar a los equipos a enfocarse en sus objetivos y ayudarlos a evitar ser expulsados de la vía por otras preocupaciones menos importantes.

A la petición de Object Management Group (OMG), Jeff Sutherland y Ken Schwaber se reunieron para resumir lo que ellos sabían acerca del desarrollo de

[3] La palabra new se repite dos veces para enfatizar que es mas que nuevo.

[4] Eazel fue una compañía de software en California que funcionó desde 1999 al 2001.

productos. Jeff Sutherland y Ken habían tenido experiencia en el manejo de productos de software comercial.

Jeff y Ken trabajaron juntos para resumir los que ellos habían aprendido en estos años, entre tanto se encontraban métodos que ellos determinaron como los más efectivos y formaban parte de una metodología coherente. Llamando a este método Scrum.

Desde su introducción en 1995, Scrum ha sido usado para construir una amplia gama de productos incluyendo data warehouses, portales de internet, sistemas de reconocimiento de voz, productos de telemedicina, middleware y aún DWFM hardware de láser ópticos.

1.9.4. La naturaleza de Scrum

Si aprendiéramos solo una cosa de este método, lo más importante sería lo que Ken aprendió durante años. Ken ha invertido el tiempo implementando Scrum por sentido común, simples prácticas. En la industria del software nosotros tendemos a confiar en complicaciones, cosas sofisticadas y difíciles de hacer, intelectualmente desafiantes, algunas veces nosotros tenemos que hacer un esfuerzo para mantener las cosas simples.

Nosotros debemos mantener a Okhan's Razor en mente. **La solución más simple es a menudo la mejor solución.**

Scrum ha sido categorizada como metodología ligera, o metodología mínima. Después de todo una metodología minimalista es en esencia relativamente natural y discreta.

Las metodologías minimalistas cambian solo lo que necesita ser cambiado. Ellas dejan todo lo demás solo. Scrum es en esencia nada más que una colección de prácticas razonables, tejidas juntas con algunas reglas, un par de expresiones y una buena dosis de sentido común.

1.9.5.Roles de Scrum.

Scrum define varios roles, los cuales están divididos en 2 grupos que son cerdos y gallinas.

Los 2 grupos vienen inspirados en la siguiente Historia:

Figura 2: *Ilustración de Roles de Scrum.*
Fuente: www.implementingscrum.com/
Elaborado por: Clark & Vizdos. **Traducido** por Darshan Zamora

"En un plato de huevos con tocino el cerdo está comprometido, la gallina sólo está involucrada".

De este modo los cerdos son los que están comprometidos en el desarrollo del software regularmente mientras que las gallinas, que son el resto del equipo, son los interesados en el proyecto. En el caso de que el proyecto falle los cerdos serán los que tienen las de perder y no las gallinas.

Definiciones

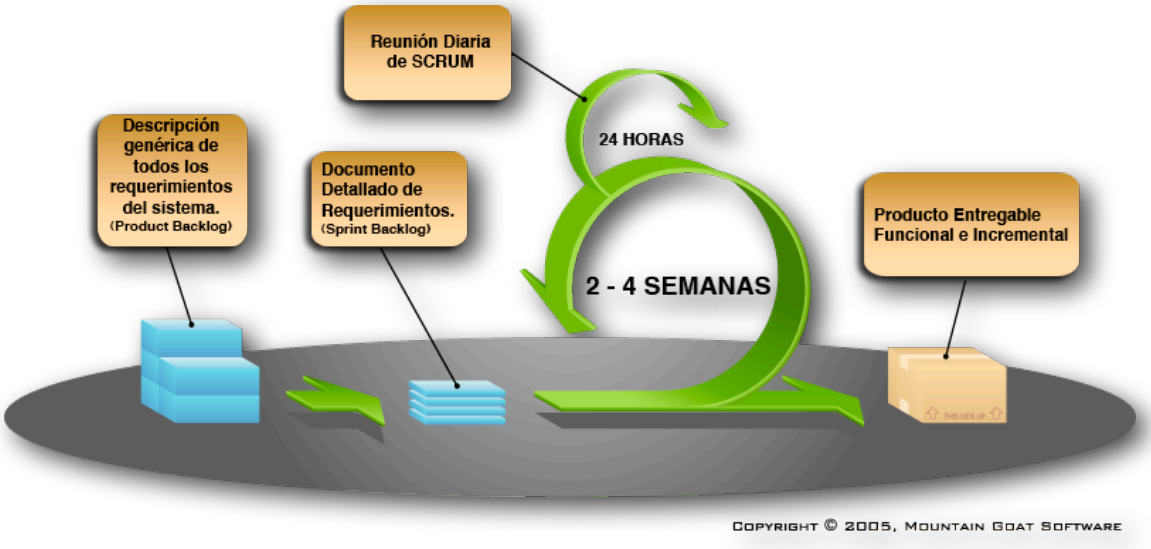


Figura 3:*Diagrama del funcionamiento de Scrum.*
Fuente: http://www.mountaingoatsoftware.com/scrum/overview
Elaborado por: Mountain Goat Software **Traducido** por Darshan Zamora

Scrum meeting: Una reunión de 15 minutos diarios donde el scrum team comparte el estado del proyecto e identifica los obstáculos y programa como eliminarlos.

Sprint: Una iteración de desarrollo usualmente termina en 30 días calendarios, durante la cual el equipo incrementalmente y demostrablemente añade funcionalidad al producto.

Backlog: El desarrollo de productos es manejado por el backlog de un producto. El backlog consiste en una lista de características, facilidades, contenidos, y tecnología que los desarrolladores han planeado incorporar en el producto. El administrador del producto mantiene esta lista y determina la prioridad de los ítems contenidos en ella. La prioridad de cada ítem es importante. Así, como los ítems que tienen la prioridad más alta deben ser abordados antes que los que tienen baja prioridad.

Antes de un Sprint el equipo se reúne con el administrador del producto y con las demás partes interesadas. Basados en la actual funcionalidad del producto y los requerimientos del mercado, el cliente y el equipo seleccionan del backlog los ítems del producto que el equipo cree que puede lograr durante el siguiente Sprint. El administrador del producto tiene la última palabra sobre cual ítem se va abordar y el equipo tiene la última palabra sobre cuantos ítems serán abordados.

Burn down

Es una gráfica mostrada públicamente, en la cual se mide la cantidad de requisitos en el backlog del proyecto pendiente al comienzo de cada sprint; representa el trabajo por hacer en un proyecto: en el eje vertical se muestra el backlog y en el horizontal se muestra el tiempo. Esta gráfica nos muestra cuando se completará todo el trabajo

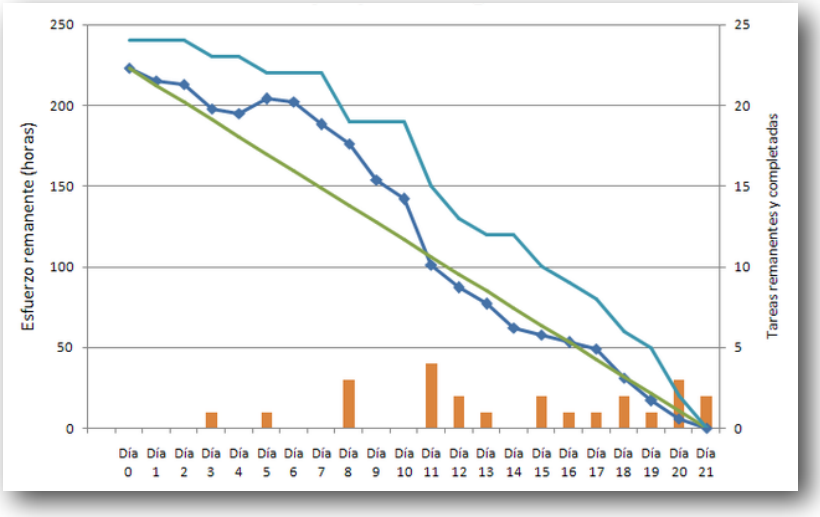


Figura 4:*Grafica Burn down.*

Fuente: http://commons.wikimedia.org/wiki/File:EjemploDeDiagramaBurnDown.png

Elaborado por: PabloStraub

1.9.6.Roles Principales

Scrum mantienen bien definidos 3 roles que son parte del “Equipo de Scrum”:

- **Scrum Máster** Facilitador
- **Scrum Product Owner** Dueño del Producto
- **Team** Equipo de Desarrollo

Scrum Máster, es el Facilitador del Equipo, esta persona es la responsable de asegurar que el ScrumTeam ejecute las prácticas, reglas y valores que el método propone, se encarga de que las reuniones se realicen en el tiempo acordado y que cada miembro del equipo tenga lo que necesita para hacer un trabajo exitoso. Si el equipo enfrenta una circunstancia difícil o un asunto que detiene el progreso del equipo, el ScrumMáster es el responsable de obtener una solución rápida.
El ScrumMáster no es el líder del equipo ya que ellos se se auto-organizan.

Scrum Product Owner, Dueño del Producto es la persona que se encarga del Product Backlog (una lista de actividades que se espera realizar) ordena los ítems por prioridad y define las condiciones de aprobación y es el responsable de todas las cosas que salen del lanzamiento del producto. Representa la voz del cliente.

Team, el equipo es un grupo de 3 a 9 personas multi-funcionales que se auto-organiza y auto-gestiona. Lo conforman personas de análisis, diseño, desarrollo, pruebas, documentación, Bases de Datos, redes y toda persona que el equipo necesite para entregar el producto descrito en Product Backlog. Entonces selecciona el Product Backlog y lo convierte en un producto terminado.

El Scrum Team, es un equipo interfuncional de 5 a 8 personas que está conformado por el Product Owner, el Scrum Máster y el Team.
Este es responsable de tornar el proyecto en el Product Backlog e investigar la forma de como entregar un pequeño demostrable incremento del proyecto en cada Sprint.
Los miembros del equipo son responsables colectivamente por el éxito de cada parte y del proyecto completo.

1.9.7. Reuniones en Scrum

Daily Scrum es una pequeña reunión que se realiza todos los días, en la cual los miembros del equipo reportan el estado del proyecto así como los obstáculos que se han encontrado. En esta reunión se empieza con la retroalimentación del método así como la evaluación del equipo y su administración .

Las reuniones diarias nos muestran el progreso que ha tenido el proyecto y entrega una idea de la oportunidad que tiene el proyecto de ejecutarse con éxito. El Daily Scrum está comandado por las siguientes reglas:

- Las reuniones empiezan a la hora escogida y hay castigos para quien llega tarde como dinero, ejercicios, llevar colgada una gallina en el cuello, entre otras.
- Todos son bienvenidos pero solo los cerdos pueden hablar.
- La reunión tiene una duración de 15 minutos y no es definida por el número de integrantes en la reunión.
- Todos los asistentes deben mantenerse de pie, lo que garantiza una reunión corta.
- La reunión se realizará todos los días a la misma hora y en el mismo lugar.
- Durante cada una de las reuniones los miembros deben contestar 3 preguntas.
  - **¿Que has hecho desde ayer?**
  - **¿Que es lo que planeas hacer hoy?**
  - **¿Has tenido algún problema que te haya impedido alcanzar tu objetivo?**

Scrum de Scrum

Después del “Daily Scrum” se reúnen las personas encargadas de cada equipo y se enfocan en las áreas de solapamiento e integración. Además de las preguntas del Daily scrum, se plantean las siguientes.

- **¿Qué ha hecho tu equipo desde nuestra última reunión?**
- **¿Qué hará tu equipo antes de que nos volvamos a reunir?**
- **¿Hay algo que demora o estorba a tu equipo?**
- **¿Estás a punto de poner algo en camino del otro equipo?**

Sprint Planning Meeting (Reunión de planificación del Sprint)

Para planificar un nuevo ciclo de Sprint, se realiza una reunión de 8 horas de duración para Sprints de 30 días y de 4 horas para Sprints de 15 días. En esta reunión se seleccionará el trabajo que se realizará. Esta reunión se divide en 2 partes. En la primera parte el dueño del producto menciona los requerimientos y objetivos como relatos de los usuarios para decidir con qué equipo de retroalimentación se trabajará.

La primera parte de la reunión se enfoca principalmente a contestar la pregunta Qué?

En la segunda parte de la reunión se enfoca en el cómo. El equipo de desarrollo intentará identificar las tareas que se relataron previamente por el dueño del producto y deducir cuanto tiempo en horas tomará realizarlas. A menos que el equipo use alguna clase de software de planeación, se puede considerar un tablero de tareas, este será alguna clase de pizarra o pared para que el equipo pueda fácilmente localizar el seguimiento del proyecto.

Una vez terminada la reunión para planificar el sprint, empieza el daily Scrum o Daily Standup estas reuniones pueden tomar hasta 30 minutos en un inicio pero con el tiempo se debe ir reduciendo más y más hasta llegar alrededor de 15 minutos.

Cada Sprint debe ser de 1 a 4 semanas

Sprint Retrospective (Retrospectiva del Sprint)

Después de terminar un Sprint y antes de iniciar uno nuevo, se reúnen en un sprint retrospective para identificar que funcionó y que no funcionó en el actual Sprint. El objetivo es ver cómo ello puede hacer la colaboración aún más efectiva para iniciar un nuevo Sprint. El tiempo que se da a estas reuniones es de 3 horas para un sprint de 30 dias pero el tiempo debe ser proporcional para el tiempo del sprint.

1.9.8. Por qué Ágile y Scrum son efectivos en proyectos de software?

Aún cuando Ágile y especialmente Scrum puede ser difícil de implementar, éste ha sido probado y ha sido extremadamente efectivo cuando se le desarrolla apropiadamente.
Aquí mencionamos 4 ventajas que podemos enfatizar.

- Una sistemática reducción del mecanismo de riesgo. Cada profesional que es responsable de la planificación y de la ejecución de un proyecto, debe conocer cuán importante es reducir el nivel de riesgo o la incertidumbre a cero, o lo más bajo posible.
- Un desarrollo de software con un ciclo de vida más ágil: Usando un ciclo de vida más ágil, hacemos más eficiente el uso del tiempo productivo.
- Una mejor adaptividad al manejo de los procesos del proyecto. A diferencia del proceso secuencial usado en el ambiente de cascada, el cual considera la estabilidad, Scrum puede verse más como que los cambios son considerados las únicas constantes.
- El manejo de un proyecto y el proceso de desarrollo del framework basado en la motivacion de la gente y el orgullo. Más que ninguna cosa entonces, este puede ser uno de los más poderosos principios de Scrum. El nuevo enfoque no está teniendo un administrador ordenando, a los miembros del equipo, lo que deben estar haciendo, pero les permite al equipo decidir por si mismo cómo se podría cumplir con su trabajo.

Scrum propone un nuevo framework de administracion de software, el cual es basado en un proyecto auto organizado del equipo, motivación, propiedad y orgulloso de sus logros.

CAPÍTULO 2

En este capítulo se detallan los antecedentes de Acción social de la PUCE de manera que nos prepara una antesala a la actividad que se realiza, observando una pequeña muestra de las maravillosas actividades que se emprenden para la comunidad que tanto necesita.

2. Análisis de la situación Actual de Proyectos de Acción Social PUCE

2.1.Antecedentes

La PUCE y en sí la Facultad de Ingeniería, aprovechando sus conocimientos, han decido ayudar a gente necesitada desde el año 1984 con estupendos resultados. Con el paso del tiempo y la evolución de la tecnología, hemos logrado superar ciertas barreras como la reducción de costos y tiempo.

Aún con todos estos avances, hemos visto proyectos truncados por falta de recursos económicos. De tal modo que la acción social ha pasado a ser solo cuestión de servicio con tiempo y persona: es decir, se ha dedicado solamente a ayudar con mano de obra a personas necesitadas, dando clases a niños, ayudando ancianos y donando un poco de víveres y ropa que ciertamente no deja de ser una ayuda excepcional.

Pero la acción social en la PUCE está superando esta barrera. Existen proyectos de construcción de escuelas, centros de cómputo, centros de salud que con ayuda de alumnos y profesores se han realizado. Planos, estudios de terreno, Contamos con un voluntariado idóneo dispuesto a dar cátedras en esas escuelas, estudiantes y profesores de medicina capaces de atender en esos centros de salud. Y prestos a construir estas obras.

Al no existir los recursos económicos necesarios para poner en marcha estos proyectos, se han hecho carreras, rifas, auspicio para soportar estas propuestas, pero no ha sido suficiente, quedando en papeles y medios digitales muchos de ellos.

He aquí la necesidad de tener una herramienta que nos permita buscar ayuda en fundaciones y organizaciones, que, de algún modo u otro, pueden financiar estos muchos proyectos listos para ponerlos en marcha.

2.2.Actualidad

En la actualidad, Acción Social de la Facultad de Ingeniería, mantiene viva esa visión de responder a las necesidades de transformación y desarrollo social, y continúa contribuyendo con justicia, solidaridad y equidad social a los grupos más vulnerables de nuestro país.

A continuación se describen los proyectos actuales que están siendo desarrollados por parte de Acción Social de la Facultad de Ingeniería. Cabe destacar que éstos son solo una muestra de lo que se está haciendo hasta mayo del 2013. Hay muchos proyectos ya entregados y funcionando, existen otros que están en espera y algunos más a cargo de las demás facultades.

2.3. Proyectos hasta mayo del 2013

Santa Rosa Ayora

En Santa Rosa Ayora Cayambe. Se está construyendo una guardería que beneficiará a unas 300 familias indígenas y tendrá una construcción anti-sísmica de 120 m^2 con proyección a un segundo piso con 80 m^2.

La Facultad de ingeniería civil está apoyando al proyecto con:

1. La asistencia técnica.
2. Topografía del terreno
3. Evaluación de materiales
4. Costos
5. Uso del laboratorio de resistencia de materiales

El proyecto obtuvo el financiamiento por parte de la Organización Española Ingenieros sin Fronteras con un costo total de **USD $50 000.** El proyecto se inició en diciembre del 2011 y se entregó en marzo del 2012

FINE

En FINE Pomasqui, se propone la creación de una sala de audio y video para 50 personas con capacidades especiales. Esta sala tendrá una construcción de 300 m2

La Facultad de ingeniería civil está apoyando al proyecto con:

1. Planos Arquitectónicos y estructurales.
2. Mano de Obra.

El proyecto espera el financiamiento por parte de la Embajada de Japón, con un costo total de **USD $140 000.** La embajada exigió que los costos sean calculados por un contratista externo a la Universidad.

Toca de Asís

En el barrio de la Tola, en el centro de Quito, funciona una casa que es el hogar de 20 ancianos recogidos de la calle con problemas de alcoholismo, psiquiátricos y en la pobreza extrema. La casa está en comodato por 10 años.

La Facultad de ingeniería civil está apoyando al proyecto con:

1. Diseño del ascensor.
2. Rampas de acceso.
3. Reconstrucción Física del Hogar.
4. Cálculo de Costos del proyecto.

El coordinador de Toca de Asís está buscando el financiamiento del proyecto que es de USD $12 000 para el ascensor, las rampas y USD $30.000 para la reconstrucción física, con un total USD $42.000.

Hogar Corazón de María

En la Av. De la Prensa y Luis Tufiño funciona el Hogar Corazón de María, que provee de un hogar para 300 ancianos.

La Facultad de Ingeniería en Sistemas apoyo a este proyecto con:

1. Desarrollo Implementación y Capacitación de un Sistema de Fichas médicas, psicológicas y sociológicas.
2. Donación de 5 computadores y su infraestructura de red para el funcionamiento del Sistema.

El proyecto tuvo un costo cero, ya que el sistema fue desarrollado en su totalidad por estudiantes de Ingeniería y las computadores se consiguieron como donaciones. El período de desarrollo de este proyecto fue de 2 años.

Hogar Buen Pastor

En la Av. 10 de Agosto y Ulloa está ubicado el Hogar del Buen Pastor, que alberga a 30 niños víctimas de maltrato y abuso sexual.

La Facultad de Ingeniería Civil y Sistemas está apoyando a este proyecto:

1. Reconstrucción Física de la casa.
2. Planos estructurales
3. Costos
4. Automatización de Fichas psicológicas, sociológicas y médicas

El proyecto requiere de una financiación de USD $150 000 aproximadamente para la Reconstrucción Física y la automatización de las fichas no tendrá costo, ya que será desarrollado por los estudiantes de Ingeniería en Sistemas.

Tingo Pucara

Tingo Pucara, Ambato, cuenta con una Cooperativa de Ahorro y Crédito perteneciente a una comunidad indígena que habita ahí.
La Facultad de Ingeniería en Sistemas sustentó a este proyecto:

1. Automatización del flujo de caja de la cooperativa de ahorro y crédito.

El Sistema está terminado y fue instalado el 15 de febrero del 2012; sin costo, ya que fue desarrollado por alumnos de Ingeniería en Sistemas.

Belisario Quevedo Latacunga

El proyecto Belisario Quevedo, Latacunga, buscaba desarrollar la automatización del manejo y facturación del agua de riego que beneficiaria a 10 barrios con más de 3000 habitantes.
La Facultad de Ingeniería en Sistemas apoyó a este proyecto:

1. Desarrollo, instalación y capacitación de un sistema para el manejo y facturación del agua de riego.

El Sistema está terminado, funcionando y sin costo, ya que fue desarrollado por alumnos de Ingeniería en Sistemas.

Hospital Padre Damián

El Hospital Padre Damián está ubicado en la Independencia localidad situada al sur de la Provincia de Esmeraldas.

La Facultad de Ingeniería en Sistemas apoyó a este proyecto:

1. Página web.
2. Automatización Facturación.
3. Automatización nómina.
4. Recepción de Turnos.

El módulo de nómina está terminado y entregado; los demás módulos están en espera de desarrollo.

Centro del Muchacho Trabajador

El centro del muchacho trabajador (CMT) es una organización social enfocada en la formación integral del niño trabajador y de su grupo familiar. Contribuye a la formación de personas auto-sustentables y, de esta manera, a superar su situación de extrema pobreza en muchos de los casos.

El CMT alberga a unas 2000 personas, distribuidas en 400 familias con servicios varios como alimentación, estudios, salud, entre otros.

La Facultad de Ingeniería en Sistemas apoyó a este proyecto:

1. Base de Datos Facturas de afiliados.
2. Sistemas de Notas para el Colegio CMT.
3. Sistema para la Caja de Ahorro CMT.

El módulo de Facturación ha sido entregado y está funcionando, el sistema de Notas está en pruebas y el sistema de caja de ahorro está listo para instalar. Todos los módulos han sido producidos sin ningún costo, por los estudiantes de Ingeniería.

La Unión, Atacames

La población de La Unión así como la misma ciudad de Esmeraldas requiere de suma urgencia el servicio básico de agua potable y alcantarillado.
La Facultad de Ingeniería Civil colaborará con:

1. Diseño de Conducción agua potable.
2. Alcantarillado Fluvial.
3. Alcantarillado Sanitario.

El proyecto tendrá un costo aproximado de USD $1’000 000 de dólares y se espera el financiamiento de la alcaldía de Atacames o, en su defecto, se buscará el financiamiento por otros medios.

Llano de Alba

Población de Llano de Alba Pesillo Olmedo, Cayambe.
La Facultad de ingeniería Civil colaboró con:

1. Construcción de Baños.
2. Mejora de la casa Comunal.

El proyecto fue entregado y tuvo el financiamiento de PUCETON con un costo de USD $4 000.

San Pedro y San Pablo

San Pedro y San Pablo está ubicado en la ciudad de Quito en la Vicentina Baja, en esta localidad existe un Centro de Cómputo (Cibernario) dedicado para 10 barrios
La Facultad de Ingeniería en Sistemas colabora con :

1. Clases de Computación a niños y adultos.

El proyecto obtuvo 30 computadores nuevos por autogestión y en la actualidad espera auto-sustentarse alquilando la mitad de las computadoras al público en general.

Misioneras de la Santa Trinidad

En el sur de Quito, en Cutuglagua, las misioneras de la Santa Trinidad, por medio de la recolección y clasificación de víveres sobrantes del Mercado Mayorista, (que muchas veces se encuentran en estado de descomposición), proveen la alimentación a 70 niños que viven en la miseria.
Las necesidades son:

1. Compra de un Terreno.
2. Elaboración de planos Arquitectónicos y estructurales.
3. Construcción del comedor.

El proyecto está evaluado en USD $50 000 y se encuentra en espera, tanto del financiamiento, como de la elaboración de los planos.

Oyacoto

En Oyacoto, entre Quito y Guayllabamba, yace un Hogar de niñas Shuar que posee un Centro de Cómputo con 20 computadoras donadas. Las necesidades son:

1. Profesores de computación.

El proyecto consiste en impartir clases de computación a las niñas por lo que la Facultad de Ingeniería pone a disposición alumnos para dicha finalidad sin costo agregado.

CAPÍTULO 3

En el presente capítulo se toman en cuenta los requerimientos tanto funcionales como no funcionales, mínimos para que funcione el sistema, así como las herramientas de software que se usaran. Ademas, se presentan como están agrupados los módulos del sistema.

3. Requerimientos

3.1. Requerimientos Funcionales

Los módulos a desarrollar en el sistema son los siguientes

- **Usuario Administrador.**
  - **F1 Aceso al Sistema**
    - Validar Ingreso a la aplicación.
  - **F2 Administración de Usuarios.**
    - F2.1 Ver Usuarios.
      - F2.1.1 Ver Usuarios a Detalle
      - F2.1.2 Ver Lista de Usuarios
    - F2.2 Insertar Usuario.
    - F2.3. Eliminar Usuario.
    - F2.4. Editar Usuario
  - **F3 Administración de Organizaciones.**
    - F3.1 Ver Organizaciones.
      - F3.1.1 Ver Lista de Organizaciones
      - F3.1.2 Ver Organizaciones a Detalle
    - F3.2 Insertar Organización.
    - F3.3. Editar Organización
    - F3.4. Eliminar Organización.
  - **F4 Administración de Proyectos.**
    - F4.1 Ver Proyectos.
      - F4.1.1 Ver Lista de Proyectos
      - F4.1.2 Ver Proyectos a Detalle
    - F4.2 Insertar Proyecto.
    - F4.3. Editar Proyecto
    - F4.4. Eliminar Proyecto.
  - **F5 Envío de Información.**
  - **F6 Asociar Proyectos y organizaciones.**

- **F7 Salir del Sistema.**

- **Usuario Regular.**
  - **F1 Aceso al Sistema**
    - Validar Ingreso a la aplicación.
  - **F2 Administración de Usuarios.**
    - F2.1 Ver Usuarios.
      - F2.1.1 Ver Usuarios a Detalle
    - F2.4. Editar Usuario
  - **F3 Administración de Organizaciones.**
    - F3.1 Ver Organizaciones.
      - F3.1.1 Ver Lista de Organizaciones
      - F3.1.2 Ver Organizaciones a Detalle
    - F3.2 Insertar Organización.
    - F3.3. Editar Organización
  - **F4 Administración de Proyectos.**
    - F4.1 Ver Proyectos.
      - F4.1.1 Ver Lista de Proyectos
      - F4.1.2 Ver Proyectos a Detalle
    - F4.2 Insertar Proyecto.
    - F4.3. Editar Proyecto
  - **F5 Envío de Información.**
  - **F6 Asociar Proyectos y organizaciones.**
  - **F7 Salir del Sistema.**

3.2.Requerimientos no Funcionales

Las Herramientas y software a usar en la implementación del sistema son las siguientes.

- **Apache 2.2.20 ,** como servidor web, ya que es software libre y compatible con PHP.
- **PHP 5.3.6,** como lenguaje de programación del lado del servidor también es software libre y compatible con MySQL.
- **MySql 5.0.8**, motor de base de Datos Compatible con PHP, Software libre.
- **JavaScript,** para hacer una interface gráfica más amigable al usuario.
- **Netbeans 7.0.1,** Software Libre IDE (Entorno de Desarrollo Integrado) de Programación compatible con PHP, JavaScript y jQuery.
- **MySQL Workbench,** herramienta visual de diseño de base de datos para MySQL, Software Libre

- **MySQL Query Browser,** herramienta gráfica para crear ejecutar y optimizar consultas en un ambiente gráfico
- **MySQL Administrator,** herramienta gráfica para administrar el servidor MySQL.
- **CodeIgniter,** Framework que usando PHP facilita el desarrollo de sitios web gracias a las múltiples librerías que dispone, ademas de implementar la arquitectura MVC[1] en su estructura.

El siguiente Hardware y Software de Base es usado en el desarrollo y pruebas para optimizar un correcto funcionamiento en todo ambiente.

Desarrollo			
Plataforma	**CPU**	**Memoria RAM**	**Browser**
Apple MAC OS X 10.8.3	Intel core 2 Duo 2.53 Ghz	4 Gb	Safari 6.0.4, Firefox 3.6.3
Apple MAC OS X 10.8.3	Intel core i5 2.5 Ghz	4 Gb	Safari 6.0.4, Firefox 3.6.3
Pruebas			
Windows XP SP 3	Intel Pentium 3 1 Ghz	768 Mb	Internet Explorer
Linux Ubuntu 9	Intel Pentium 3 1 Ghz	768 Mb	Firefox
Apple MAC X OS 10.6.8	Intel core 2 Duo 2.53 Ghz	4 Gb	Safari 6.0.4, Firefox 3.6.3

[1] MVC se detalle en la pagina 82

CAPÍTULO 4

El capítulo 4 nos muestra la estructura del sistema desde un punto de vista global hasta el detalle, es el medio por el cual se abstrae las vistas del sistema y se puede observar como funciona internamente así como su composición.

4.Diseño de Aplicación

4.1.Diagramas de Aplicación

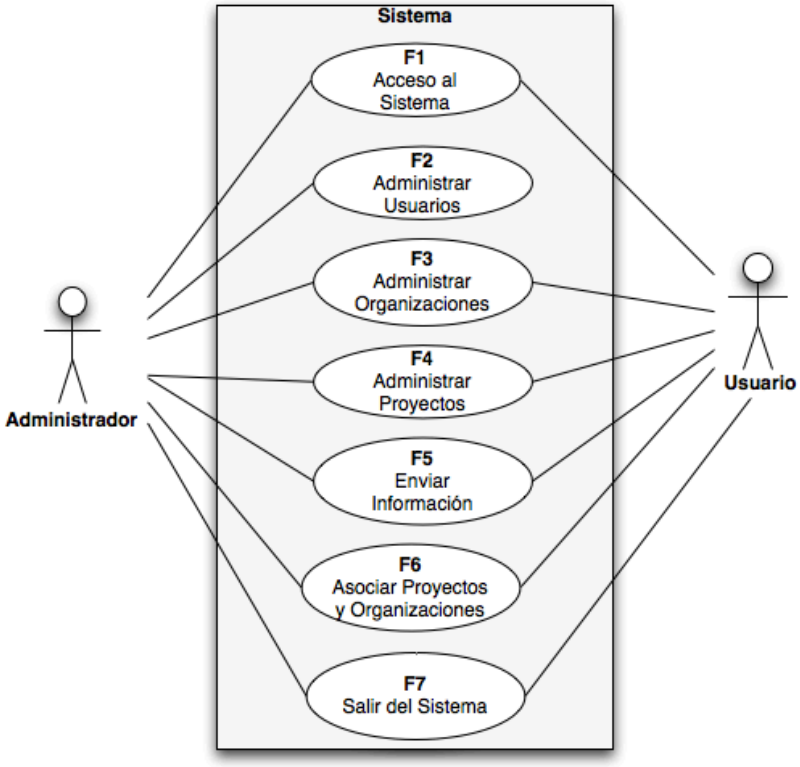


Diagrama General.

UML

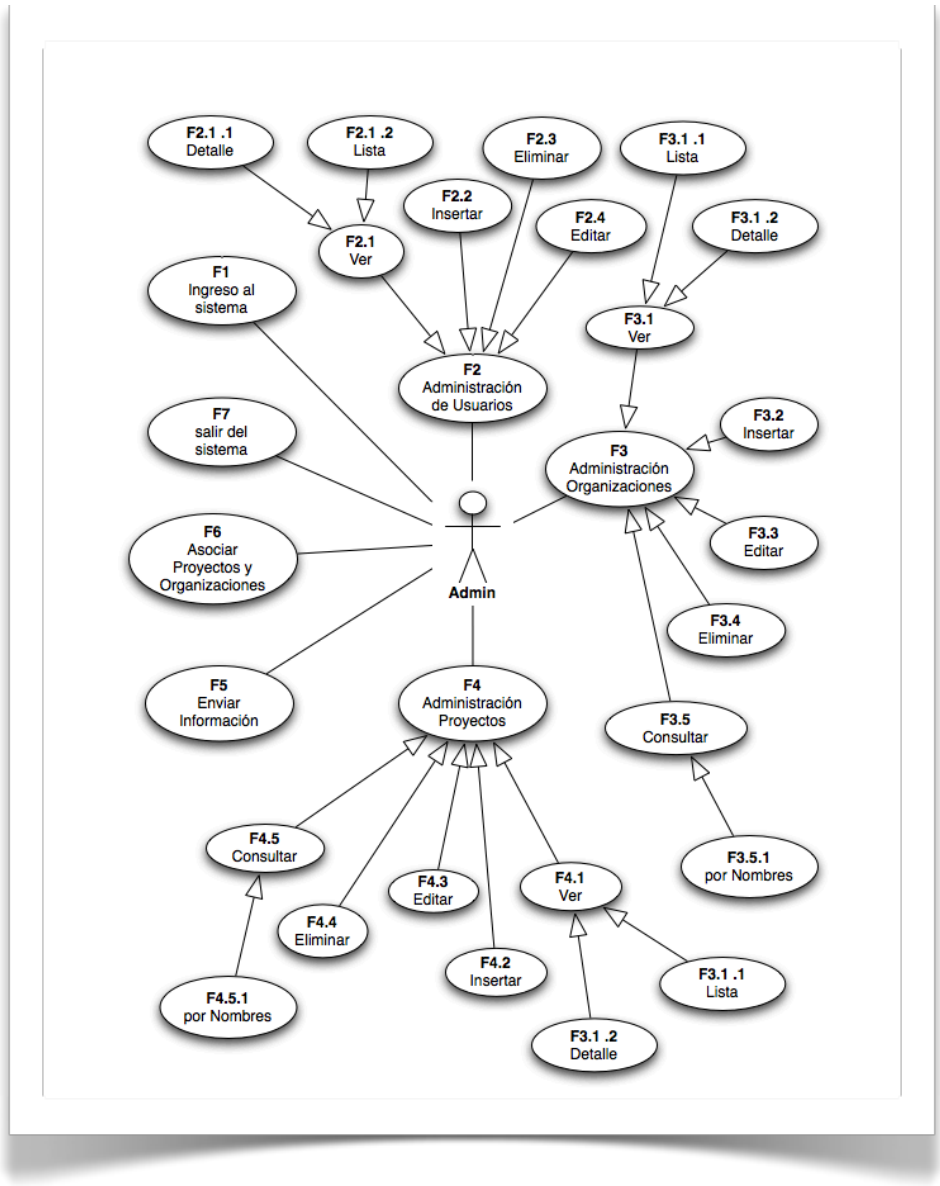
Lenguaje Unificado de Modelado es una de las herramientas más flexibles y útiles en el mundo del desarrollo de sistemas, ya que es un lenguaje de modelamiento visual que permite crear los planos del software que estamos construyendo, de modo que podamos ver de manera sencilla y fácil de entender el sistema a construir. Además es de fácil interpretación por todas las personas involucradas en la construcción del software. De esta forma, disponemos de un lenguaje fácil de interpretar y potente a la vez, para mostrar de que se trata en sí el software que deseamos construir.

Caso de Uso

Los casos de uso son una estructura que ayuda al análisis de la interacción entre los usuarios y el uso del sistema. Una colección de casos de uso, representa un sistema en términos de lo que los usuarios intentan hacer con esto.

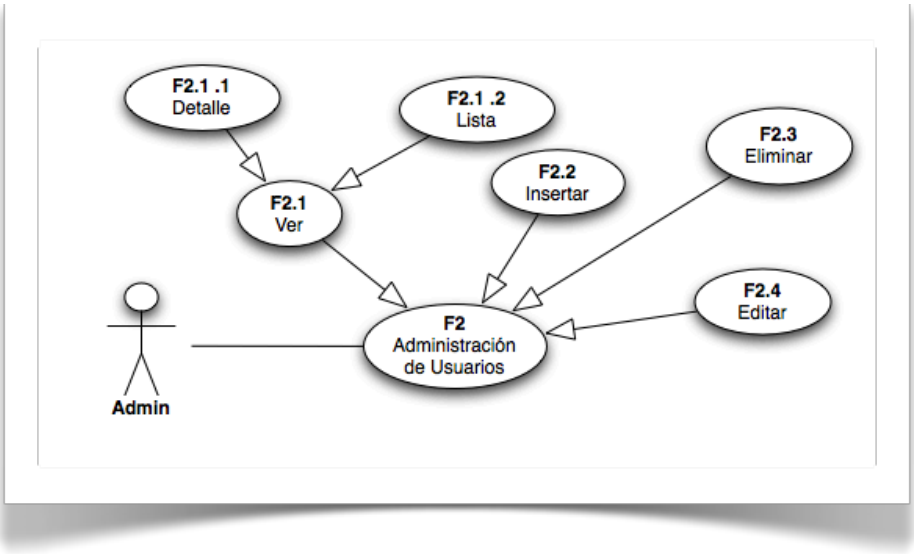
Los casos de uso son una herramienta excelente para estimular potencialmente a los usuarios en hablar acerca de un sistema desde sus propios puntos de vista, ya que no siempre es fácil para los usuarios mostrar como intentan usar un sistema .

Diagrama Casos de Uso

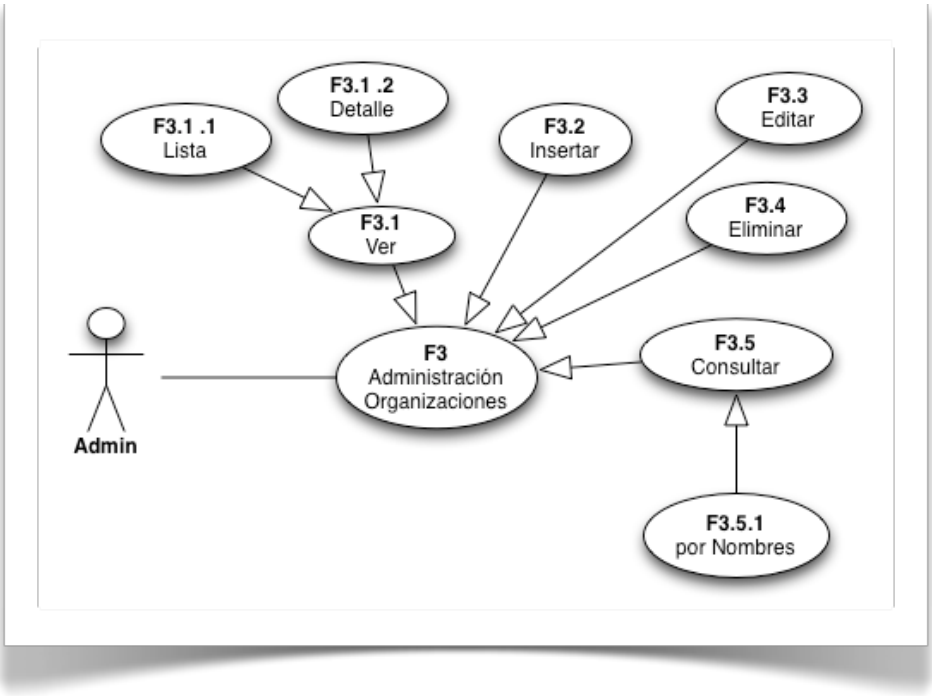


Ingreso y salida del sistema

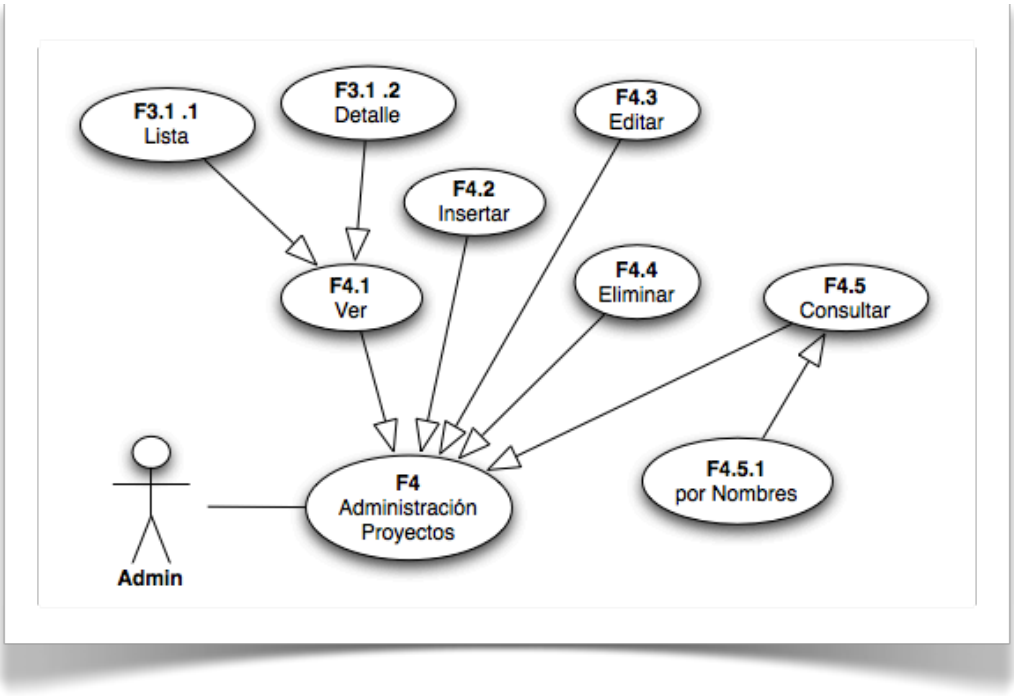
Administración de Usuarios.



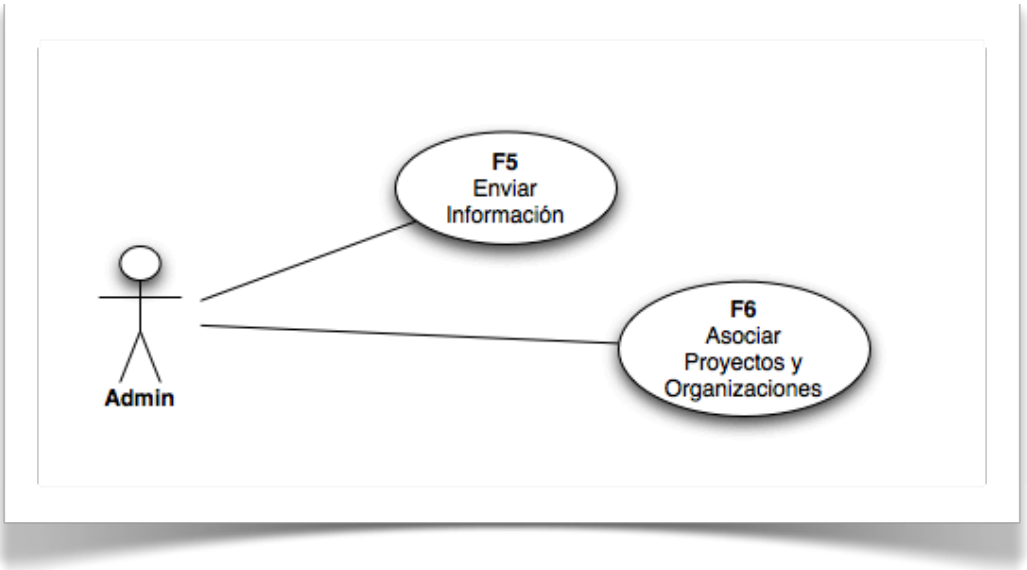
Administración de Organizaciones.



Administración de Proyectos.

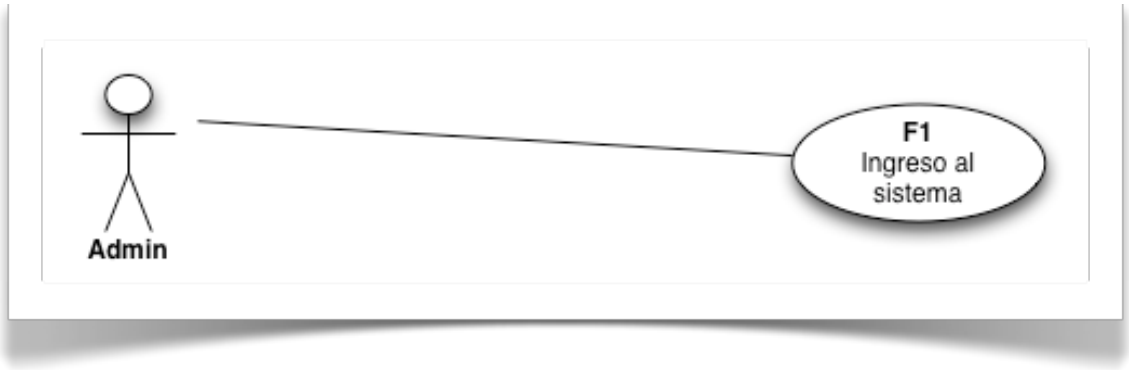


Asociación de Proyectos Organización y envío de información



Diagramas a Detalle

F 1 Ingreso al sistema



Actores: Administrador, Usuario

Descripción: En este caso de uso se valida un Administrador para permitirle ingresar al sistema como tal.

Glosario: Actor.- Debe ser administrador, ya que tendrá jerarquía sobre los usuarios, pudiendo administrarlos. Además podrá eliminar las organizaciones y proyectos.

Flujo Principal:

1. El Actor ingresa su nombre de usuario y contraseña.
2. El Actor presiona el botón OK.
3. El Sistema verifica la existencia del usuario en la BDD. (EXC1)
4. El Sistema verifica que la contraseña sea la correcta. (EXC2)
5. El sistema abre la página principal

Excepciones (EXC):

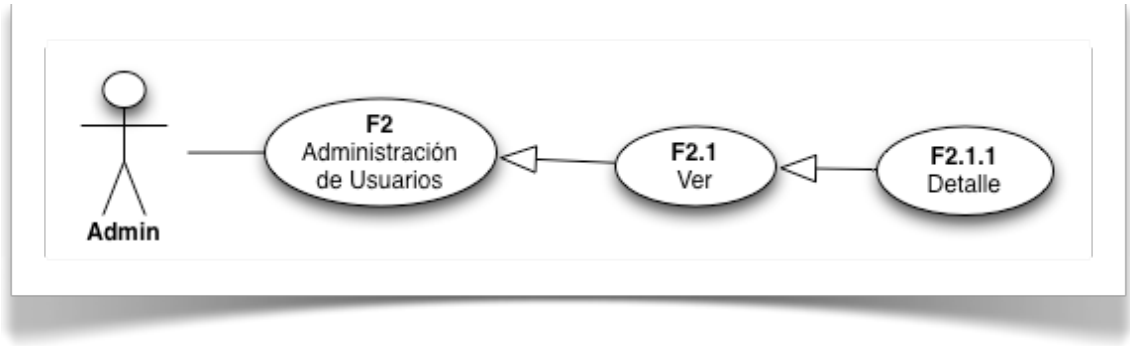
1.No existe el Actor,

Presentar pantalla (“Nombre de Usuario inválido”).

2.La contraseña no coincide.

Presentar pantalla (“Contraseña inválida”).

F 2.1.1 Ver Usuarios detalle



Actores: Administrador

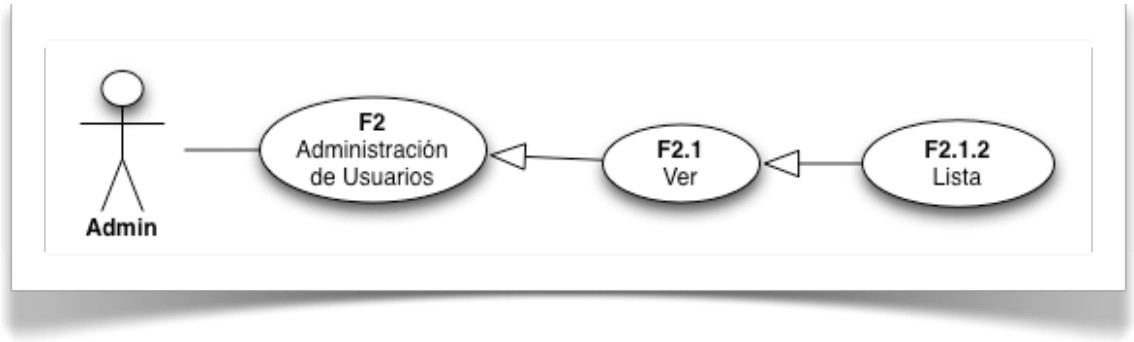
Descripción: Este caso de uso va a mostrar los datos de un usuario del sistema.

Glosario: Actor.- El administrador puede elegir ver los datos de un usuario.

Flujo Principal:

1. El Actor selecciona el nombre del usuario deseado de una lista.
2. El Sistema toma la id del usuario y lo trae de la BDD.
3. El Sistema muestra la información del usuario.

F 2.1.2 Ver Usuarios lista



Actores: Administrador

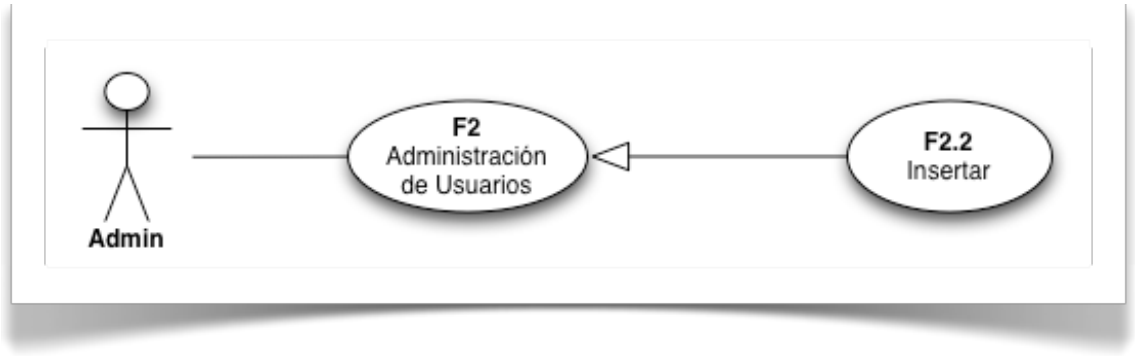
Descripción: Este caso de uso va a mostrar en una lista de todos los usuarios en el sistema.

Glosario: Actor.- El administrador, al abrir una ventana editar **F2.4** , ver **F2.1** llama a todos los usuarios en orden alfabético.

Flujo Principal:

1. El Actor abre la ventana editar **F2.4** o ver **F2.1**.
2. El Sistema trae a los usuarios de la BDD.
3. El Sistema muestra los nombres de los usuarios en orden alfabético.

F2.2 Insertar Usuarios



Actores: Administrador.

Descripción: Este caso de uso se van a insertar Usuarios al sistema.

Glosario: Actor.- El administrador, presionar el botón nuevo se muestra el formulario de nuevo usuario en blanco.

Flujo Principal:

1. El Actor pulsa el botón nuevo.
2. El Sistema muestra el formulario en blanco.
3. El Actor ingresa Cédula, Nombre, Apellido (EXC1).
4. El Actor ingresa si es o no Administrador.
5. El Actor pulsa el botón. Guardar (EXC2)
6. El Sistema ingresa la contraseña por defecto CI.
7. El Sistema guarda el nuevo usuario en la Base de Datos.
8. El Sistema actualiza la vista de lista **F2.1.2** con el nuevo usuario ingresado.

Excepciones (EXC):

1.No pasó la validación de los Datos,

Presentar en color Rojo la causa del error de validación.

2.No se pudo guardar el Usuario,

Presentar pantalla (“NO se guardó el Usuario”).

Mantiene Datos ingresados en Formulario.

F2.3 Eliminar Usuarios

Actores: Administrador.

Descripción: Este caso de uso se van a Eliminar Usuarios de la base de Datos.

Glosario: Actor.- El administrador, selecciona el usuario a eliminar y lo borra de la base de Datos .

Flujo Principal:

1. El Actor selecciona un Usuario de la lista.
2. El Sistema muestra los datos del usuario seleccionado en el formulario.
3. El sistema visualiza y pone como disponible el botón Eliminar.
4. El Actor presiona el botón Eliminar.
5. El Sistema muestra el mensaje (“Se va a eliminar a [nombre][apellido]”).
6. El Actor presiona el botón Aceptar.
7. El Sistema Borra al usuario de la base de Datos.(EXC1)
8. El Sistema actualiza la vista de lista **F2.1.2**.

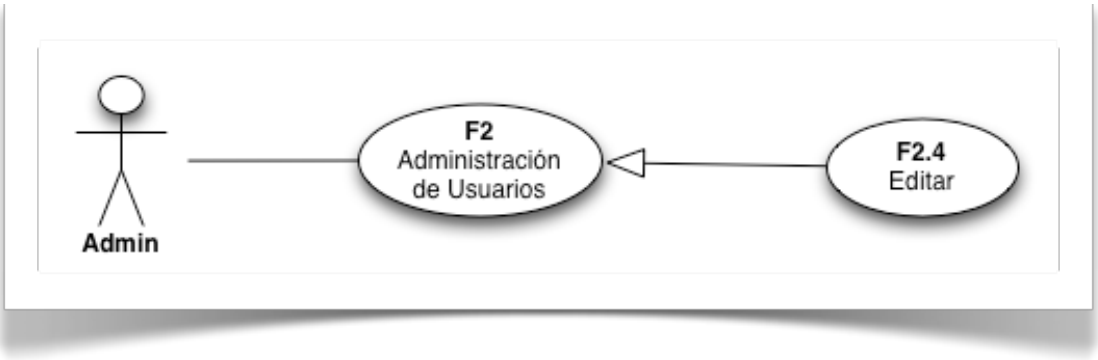
Excepciones (EXC):

1.No se pudo borrar al usuario de la base de Datos,

Presentar pantalla (“No se eliminó el Usuario”).

Mantiene Usuario seleccionado.

F 2.4 Editar Usuarios



Actores: Administrador.

Descripción: En este caso de uso se van a Editar Usuarios de la base de Datos.

Glosario: Actor.- El administrador, selecciona el usuario a editar y se actualiza la BDD.

Flujo Principal:

1. El Actor selecciona un Usuario de la lista.
2. El Sistema muestra los datos del usuario seleccionado en el formulario.
3. El sistema visualiza y pone como disponible el botón. Editar.
4. El Actor puede editar Cédula, Nombre, Apellido (EXC1).
5. El Actor puede editar si es o no Administrador.
6. El Actor pulsa el botón. Guardar (EXC2)
7. El sistema restaura la contraseña por defecto Ci.
8. El Sistema actualiza los datos en la Base.
9. El Sistema actualiza la vista de lista **F2.1.2**.

Excepciones (EXC):

1.No pasó la validación de los Datos,

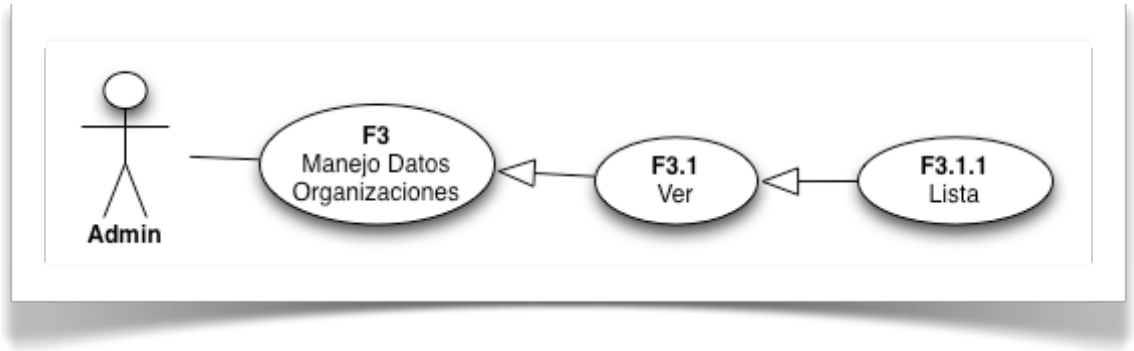
Presentar en color Rojo la causa del error de validación.

2.No se pudo guardar el Usuario,

Presentar pantalla (“NO se guardó el Usuario”).

Mantiene Datos ingresados en Formulario.

F 3.1.1 Ver Organización lista



Actores: Administrador, Usuario

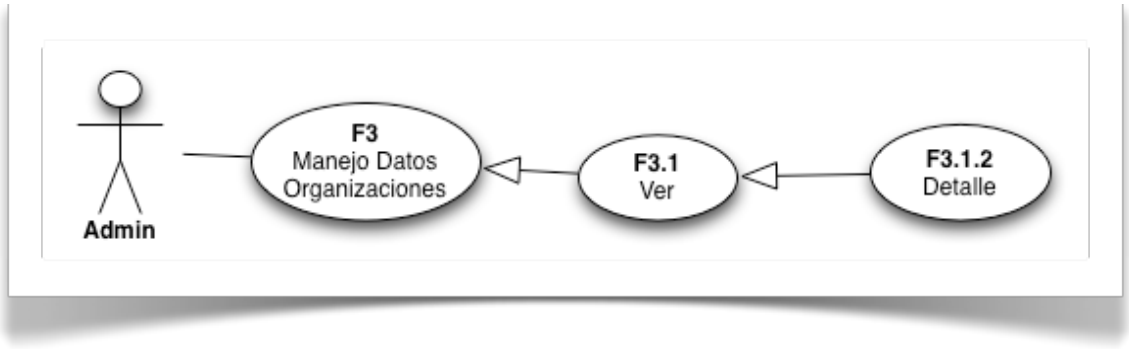
Descripción: Este caso de uso se va a mostrar en una lista de todos las Organizaciones.

Glosario: Actor.- El administrador puede ver a todos las Organizaciones. Usuario puede ver solo las que estén activas.

Flujo Principal:

1. El Sistema abre la ventana Principal.
2. El Sistema trae las organizaciones de la BDD.
3. El Sistema muestra los nombres de las organizaciones.

F 3.1.2 Ver Organizaciones detalle



Actores: Administrador, Usuario

Descripción: En este caso de uso se van a mostrar los datos a detalle de una Organización.

Glosario:

Actor.- Puede elegir ver los datos de una Organización.

Flujo Principal:

1. El Actor selecciona el nombre de la Organización de la lista.
2. El Sistema toma la id de la Organización y lo trae de la BDD.
3. El Sistema muestra la información de la Organización.

F 3.2 Insertar Organización

Actores: Administrador, Usuario.

Descripción: En este caso de uso se va a insertar una Organización a la BDD.

Glosario: Actor.- El administrador tiene la capacidad de decidir el estado de una organización como activa o no.

Flujo Principal:

1. El Actor pulsa el botón nuevo.
2. El Sistema muestra el formulario de Organización en blanco.
3. El Actor ingresa los datos de la Organización (EXC1).
4. El Actor pulsa el botón. Guardar (EXC2)
5. El Sistema guarda el nuevo usuario en la Base de Datos.
6. El Sistema actualiza la vista de lista **F3.1.2** con el nuevo usuario ingresado.

Excepciones (EXC):

1.No pasó la validación de los Datos,

Presentar en color Rojo la causa del error de validación.

1.No se pudo guardar la Organización

Presentar pantalla (“NO se guardó la Organización”).

Mantiene Datos ingresados en Formulario.

F 3.3 Editar Organizaciones.

Actores: Administrador, Usuario.

Descripción: En este caso de uso se va a Editar una Organización de la BDD.

Glosario: Actor.- El administrador, selecciona la Organización a editar y actualiza de la BDD.

Flujo Principal:

1. El Actor selecciona una Organización de la lista.
2. El Sistema muestra los datos de la organización seleccionado en el formulario.
3. El sistema visualiza y pone como disponible el botón. Editar.
4. El Actor puede editar los datos de la Organización (EXC1).
5. El Actor puede editar el estado si es o no Activo.
6. El Actor pulsa el botón. Guardar (EXC2)
7. El Sistema actualiza los datos en la Base.
8. El Sistema actualiza la vista de lista **F2.1.2**.

Excepciones (EXC):

1.No pasó la validación de los Datos,

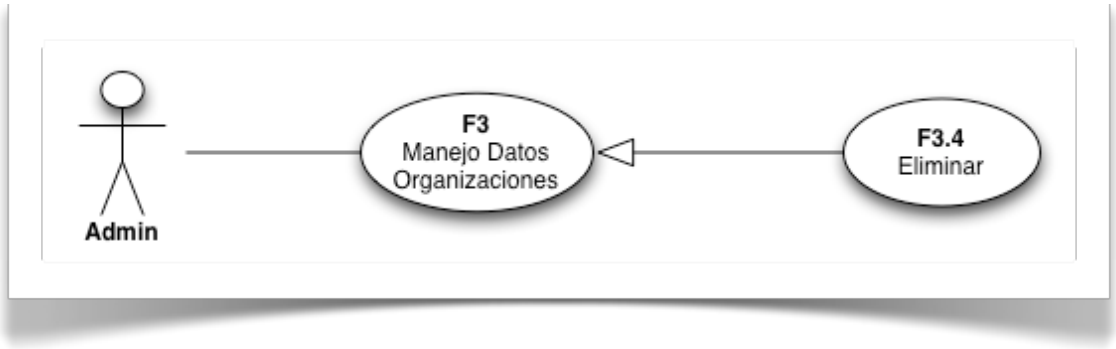
Presentar en color Rojo la causa del error de validación.

2.No se pudo guardar el Usuario,

Presentar pantalla (“NO se guardó la Organización”).

Mantiene Datos ingresados en Formulario.

F 3.4 Eliminar Organizaciones



Actores: Administrador, Usuario.

Descripción: En este caso de uso se van a Eliminar Organizaciones de la base de Datos.

Glosario: Actor.- El administrador, selecciona la organización a eliminar y lo borra de la base de Datos. El usuario realiza el mismo flujo pero solamente se pone la organización como inactiva.

Flujo Principal:

1. El Actor selecciona una Organización de la lista.
2. El Sistema muestra los datos de la organización seleccionada en el formulario.
3. El sistema visualiza y pone como disponible el botón. Eliminar.
4. El Actor Presiona el botón Eliminar.
5. El Sistema muestra el mensaje (“Se va a eliminar a [nombre organización]”).
6. El Actor presiona el botón Aceptar.
6. El Sistema Borra la Organización de la base de Datos.(EXC1)
7. El Sistema actualiza la vista de lista **F3.1.2**.

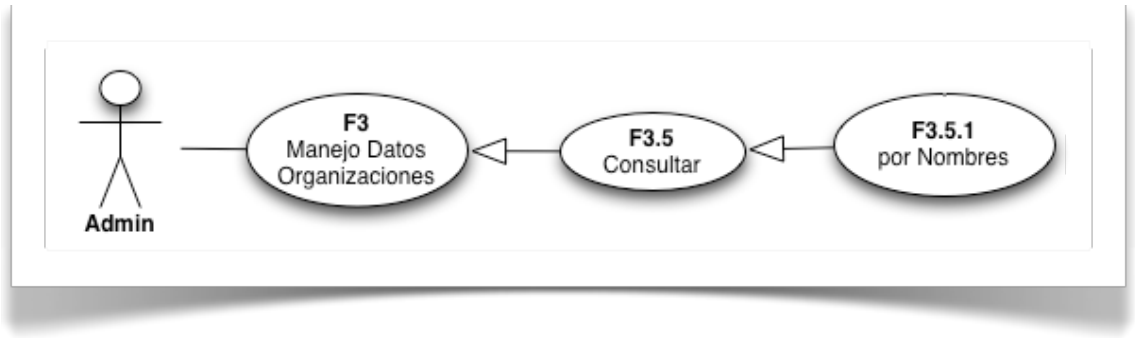
Excepciones (EXC):

1.No se pudo borrar la Organización de la base de Datos.

Presentar pantalla (“No se eliminó Organización”).

Mantiene Organización seleccionada.

F 3.5.1 Consultar Organizaciones por nombre



Actores: Administrador, Usuario.
Descripción: En este caso de uso se van a ver Organizaciones que correspondan al nombre ingresado.
Glosario: Actor.- Podrá ver todas las organizaciones que correspondan con entradas que ingresen en el campo buscar.

Flujo Principal:

1. El Actor ingresa el texto que desea buscar en BDD.
2. El sistema busca las organizaciones que tienen relación con el texto ingresado en el campo de búsqueda.
3. El Sistema muestra los datos de las Organizaciones encontradas en la lista (EXC1).

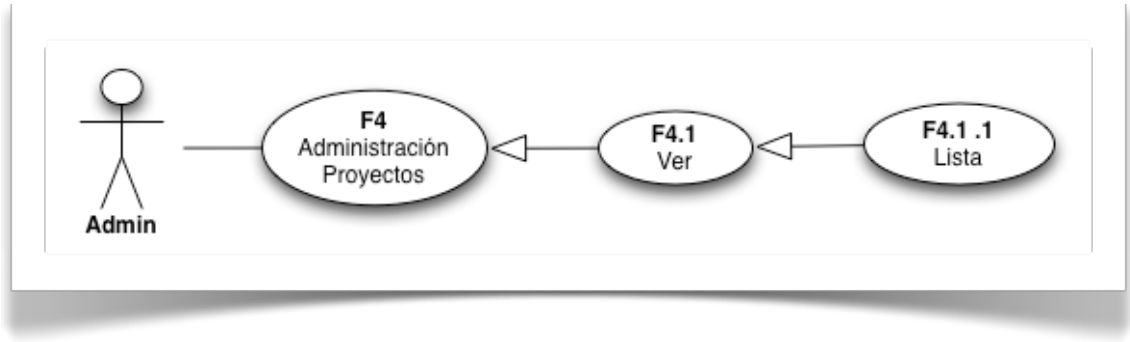
Excepciones (EXC):

1.No se encontró ninguna organización relacionada con la entrada,

Presentar pantalla (“No se encontró ningún Registro”).

Mostrar la lista de Organizaciones y el formulario en blanco.

F 4.1.1 Ver Proyecto lista



Actores: Administrador, Usuario

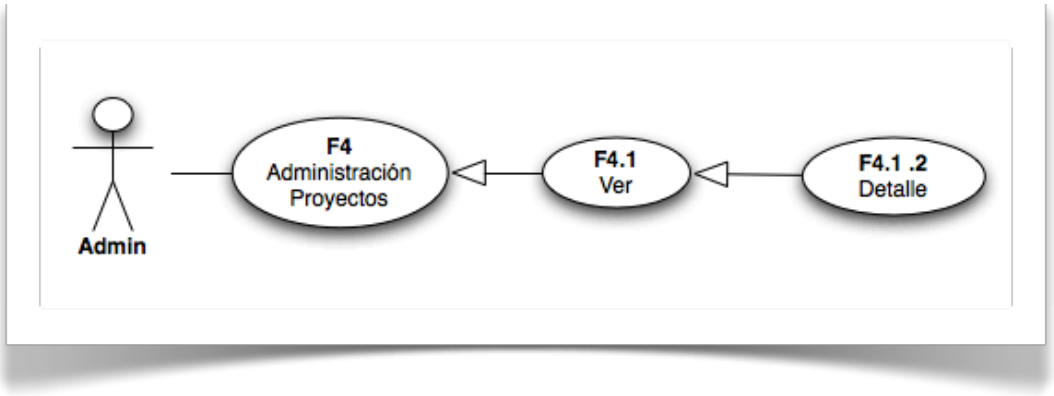
Descripción: Este caso de uso se va a mostrar en una lista de todos los Proyectos.

Glosario: Actor.- El administrador puede ver a todos los proyectos activos e inactivos.

Flujo Principal:

1. El Sistema abre la ventana Principal.
2. El Sistema trae los Proyectos de la BDD.
3. El Sistema muestra los nombres de los Proyectos.

F 3.1.2 Ver Proyectos detalle



Actores: Administrador, Usuario.

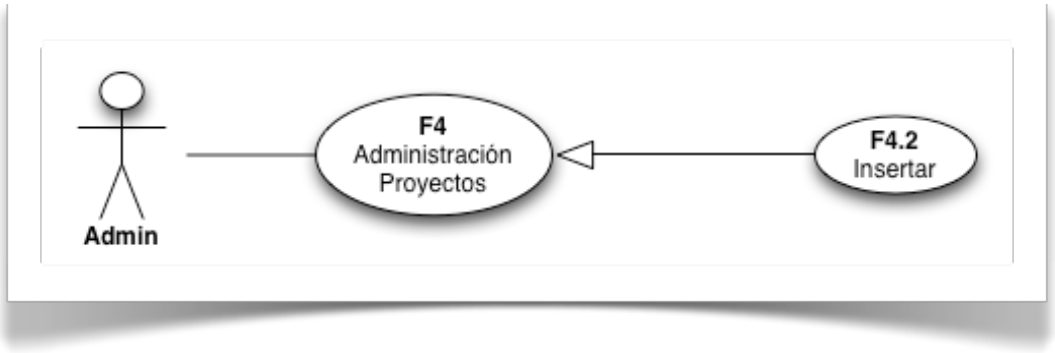
Descripción: En este caso de uso se van a mostrar los datos del proyecto a detalle.

Glosario: Actor.- Puede elegir ver los datos de un Proyecto.

Flujo Principal:

1. El Actor selecciona el nombre del Proyecto de la lista.
2. El Sistema toma la id del Proyecto y lo trae de la BDD.
3. El Sistema muestra la información del Proyecto.

F 4.2 Insertar Proyectos



Actores: Administrador, Usuario.

Descripción: En este caso de uso se va a insertar un Proyecto a la BDD.
Glosario: Actor.- El administrador esta autorizado para decidir el estado de un Proyecto como activo o no.

Flujo Principal:

1. El Actor pulsa el botón nuevo.
2. El Sistema muestra el formulario de Proyecto en blanco.
3. El Actor ingresa los datos del Proyecto (EXC1).
4. El Sistema muestra seleccionado el campo activo.
5. El Actor pulsa el botón. Guardar (EXC2)
6. El Sistema guarda el nuevo Proyecto en la Base de Datos.
7. El Sistema actualiza la vista de lista **F4.1.2** con el nuevo usuario ingresado.

Excepciones (EXC):

1.No pasó la validación de los Datos,

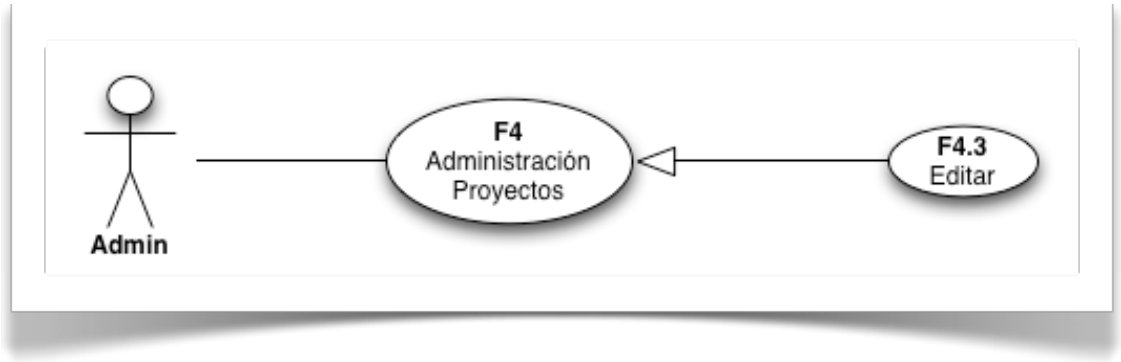
Presentar en color Rojo la causa del error de validación.

1.No se pudo guardar el Usuario,

Presentar pantalla (“NO se guardó el Proyecto”).

Mantiene Datos ingresados en Formulario.

F 4.3 Editar Proyectos.



Actores: Administrador, Usuario.

Descripción: En este caso de uso se va a Editar un Proyecto de la BDD.
Glosario: Actor.- Selecciona el Proyecto a editar y lo actualiza de la BDD.

Flujo Principal:

1. El Actor selecciona un Proyecto de la lista.
2. El Sistema muestra los datos del Proyecto seleccionado en el formulario.
3. El sistema visualiza y pone como disponible el botón. Editar.
4. El Actor puede editar los datos del Proyecto (EXC1).
5. El Actor puede editar el estado si es o no Activo.
6. El Actor pulsa el botón. Guardar (EXC2)
7. El Sistema actualiza los datos en la Base.
8. El Sistema actualiza la vista de lista **F4.1.2**.

Excepciones (EXC):

1.No pasó la validación de los Datos,

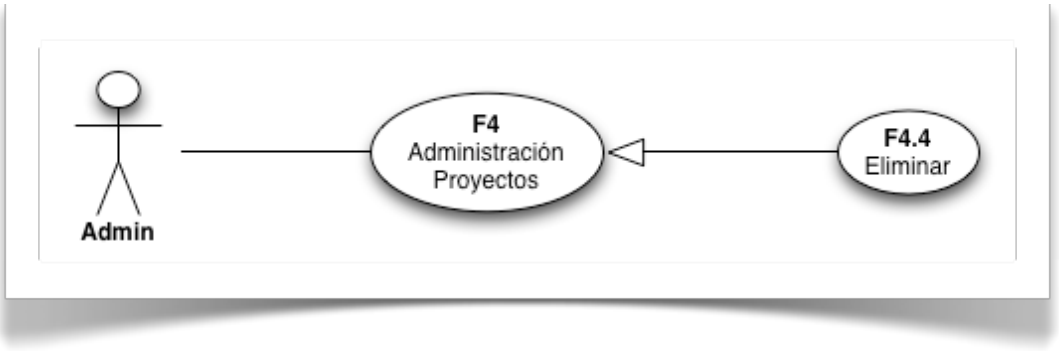
Presentar en color Rojo la causa del error de validación.

2.No se pudo guardar el Proyecto,

Presentar pantalla (“NO se guardó el Proyecto”).

Mantiene Datos ingresados en Formulario.

F 4.4 Eliminar Proyectos



Actores: Administrador, Usuario.

Descripción: En este caso de uso se van a Eliminar Proyectos de la base de Datos.

Glosario: Actor.- El administrador, selecciona el Proyecto a eliminar y lo borra de la base de Datos. El usuario solo lo desactiva siguiendo el mismo flujo

Flujo Principal:

1. El Actor selecciona un Proyecto de la lista.
2. El Sistema muestra los datos del Proyecto seleccionada en el formulario.
3. El sistema visualiza y pone como disponible el botón. Eliminar.
4. El Actor Presiona el botón Eliminar.
5. El Sistema muestra el mensaje (“Se va a eliminar a [nombre Proyecto]”).
6. El Actor presiona el botón Aceptar.
6. El Sistema Borra el Proyecto de la base de Datos.(EXC1)
7. El Sistema actualiza la vista de lista **F4.1.2**.

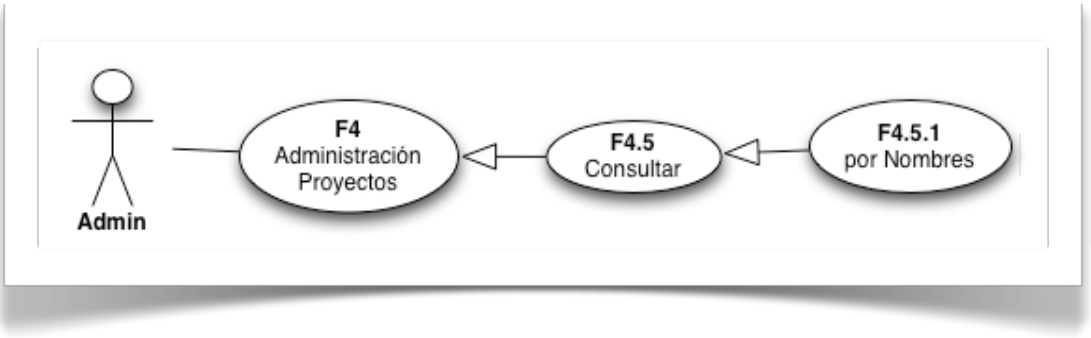
Excepciones (EXC):

1.No se pudo borrar al proyecto de la base de Datos,

Presentar pantalla (“No se eliminó Proyecto”).

Mantiene Usuario seleccionado.

F 4.5.1 Consultar Proyecto por nombre



Actores: Administrador, Usuario.

Descripción: En este caso de uso se van a ver Proyectos que correspondan al nombre ingresado.

Glosario: Actor.- Podrá ver todos los Proyectos que correspondan con entradas que ingresen en el campo buscar.

Flujo Principal:

1. El Actor ingresa el texto que desea buscar en BDD.
2. El Sistema busca Proyectos que tienen relación con el texto ingresado en el campo de búsqueda.
3. El Sistema muestra los datos del Proyecto encontrados en la lista (EXC1).

Excepciones (EXC):

1.No se encontró ninguna organización relacionada con la entrada,

Presentar pantalla (“No se encontró ningún Proyecto”).

Muestra la tabla de Proyectos y el formulario vacíos.

F 5 Enviar Información Proyecto

Actores: Administrador, Usuario.

Descripción: En este caso de uso se va a enviar la información seleccionada del proyecto seleccionado a la Organización seleccionada.

Glosario: Actor.- Escogerá la organización a la cual desea enviar la información del proyecto seleccionado. Ademas elige la información que considere relevante enviar.

Flujo Principal:

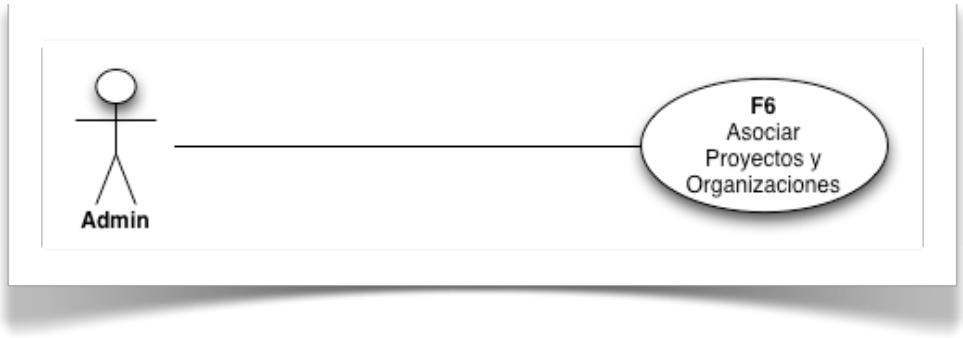
1. El Actor escoge un proyecto de la lista.
2. El sistema busca Organizaciones que tienen relación con el Proyecto elegido
3. El Actor presiona el botón enviar información.
4. El Sistema abre la ventana con los datos que se quieren enviar.
5. El Actor desactiva los checkboxes de los destinatarios que no desea enviar.
6. El Actor desactiva los checkboxes de los datos que no desea enviar.
7. El Actor presiona el botón enviar.(EXC1)

Excepciones (EXC):

1. No se Pudo enviar la información,

   Presentar pantalla (“No se envió la información”).

F 6 Asociar Proyectos y Organizaciones.



Actores: Administrador Usuario.
Descripción: En este caso de uso se van a asociar los proyectos con las organizaciones.
Glosario: Actor.- Escogerá un proyecto y el sistema generará una lista de Organizaciones que puedan soportar al proyecto elegido.

Flujo Principal:

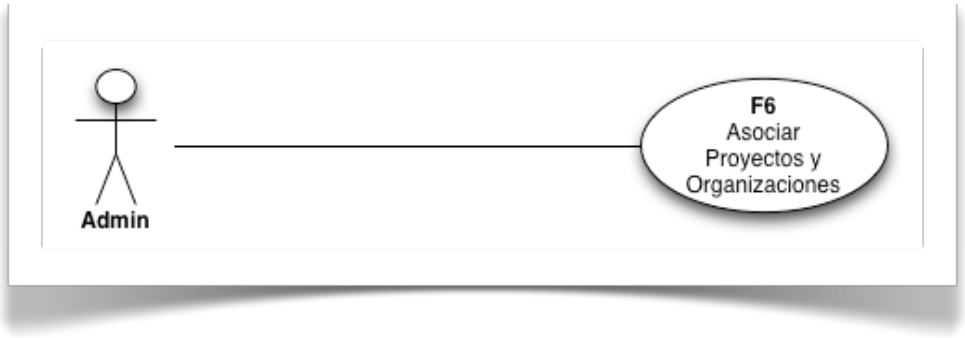
1. El Actor escoge un Proyecto la lista.
2. El Sistema busca Organizaciones relacionadas con los datos del Proyecto y los muestra en la lista a continuación. (EXC1)
3. El Sistema muestra la primera Organización de la lista en el formulario.
3. El Actor escoge una Organización de la lista.
4. El Sistema visualiza la Organización en el formulario.

Excepciones (EXC):

1.No existen coincidencias entre el Proyecto y la organización,

Presentar pantalla (“No se encontró Organización que corresponda”).

F 6 Asociar Organizaciones y Proyectos.



Actores: Administrador, Usuario.

Descripción: En este caso de uso se van a asociar las organizaciones con los proyectos.
Glosario: Actor.- Escogerá una Organización y el sistema generará una lista de Proyectos que pueden ser soportados por la Organización elegida.

Flujo Principal:

1. El Actor escoge una Organización de la lista.
2. El Sistema busca Proyectos relacionadas con los datos de la Organización y los muestra en la lista a continuación. (EXC1)
3. El Sistema muestra el primer Proyecto de la lista en el formulario.
3. El Actor escoge un Proyecto de la lista.
4. El Sistema visualiza el Proyecto en el formulario.

Excepciones (EXC):

1.No existen coincidencias entre el Proyecto y la organización,
Presentar pantalla (“No se encontró Organización que corresponda”).

F 7 Salida del sistema

Actores: Administrador, Usuario

Descripción: En este caso de uso se va salir del sistema cerrando todas las variables de sesión

Glosario: Actor.- Salir del sistema implica cerrar todas las variables de sesión, evitando que con nuestro id la gente pueda registrar usuario u organizaciones en la base de datos; así como también cerrar los accesos que son solamente del administrador.

Flujo Principal:

1. El Actor pulsa el botón cerrar sesión.
2. El Sistema cierra y borra todas las variables de sesión.
3. El Sistema se desconecta de la base de datos BDD.
4. El Sistema abre la página de validación de usuarios.

Diagrama de Actividades.

Los diagramas de actividades son muy parecidos a los diagramas de flujo. Estos muestran los pasos así como los puntos y las ramas de decisión. Es de mucha ayuda para mostrar que pasa en un proceso del negocio o una operación.

Un diagrama de actividades es diseñado para simplificar la vista de lo que estaá pasando durante una operación o proceso.

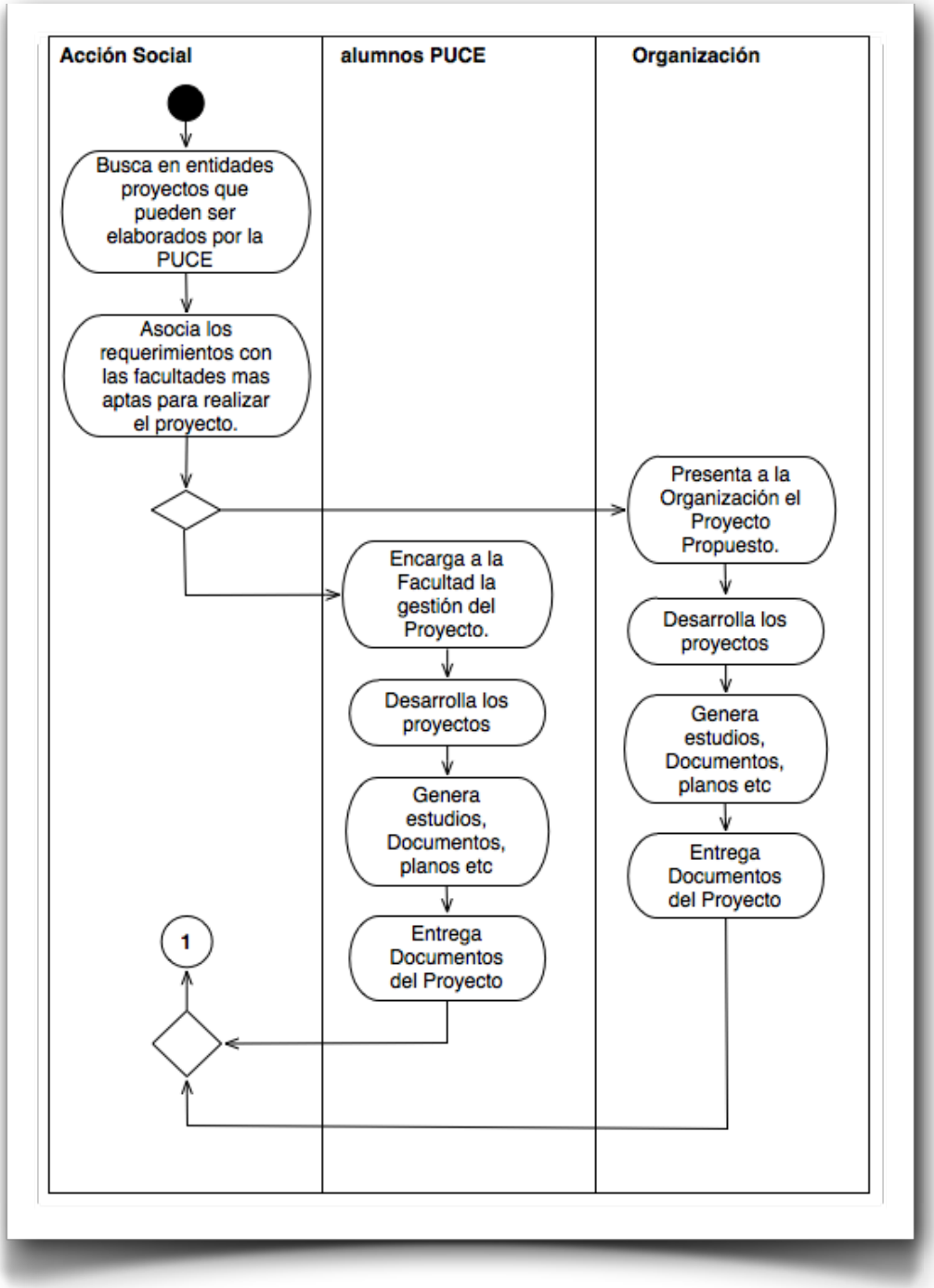
Cada actividad es representada por un rectángulo redondeado. Una flecha representa una transición de una actividad a otra; además tienen un punto de partida y un punto de llegada.

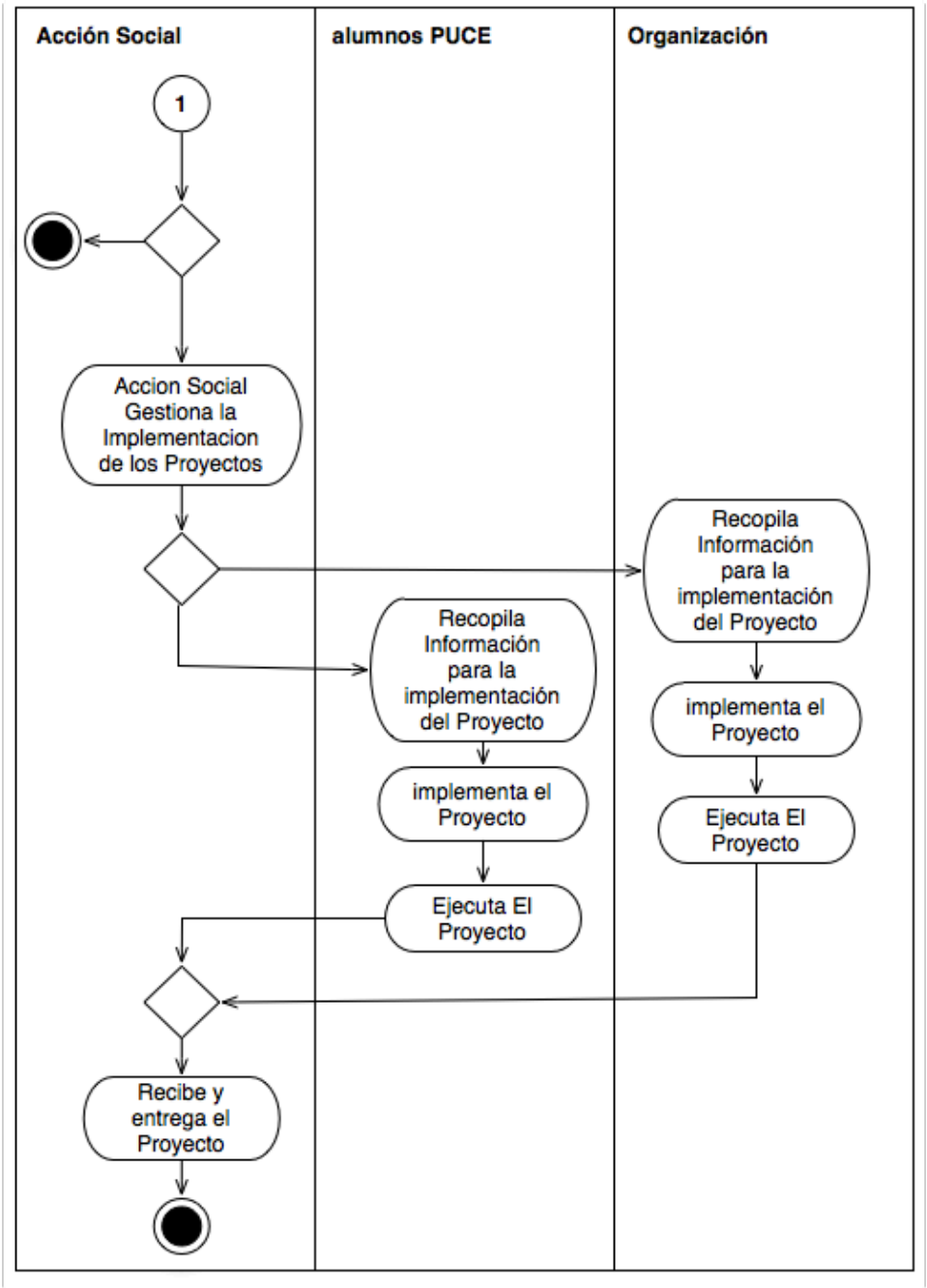
Una secuencia de actividades casi siempre tiene un punto donde se debe tomar una decisión en la cual se debe proceder de un modo o de otro. En este diagrama se representa una decisión como un diamante similar al que usa los diagramas de flujo.

Los caminos concurrentes son usualmente usados en los diagramas de actividad, estos se presentan cuando se realicen actividades paralelas en el tiempo.

Una expansión del diagrama de actividades que resulta muy práctica es la inclusión de swimlanes o carriles (similares a los usados en las piscinas olímpicas) que nos muestra quién es el responsable de cada actividad en un proceso.

Diagrama de Actividades del sistema




Acción Social
alumnos PUCE
Organización
1
Accion Social Gestiona la Implementacion de los Proyectos
Recopila Información para la implementación del Proyecto
Recopila Información para la implementación del Proyecto
implementa el Proyecto
Ejecuta El Proyecto
implementa el Proyecto
Ejecuta El Proyecto
Recibe y entrega el Proyecto

En el diagrama anterior se muestra el modo en que opera Acción social en un diagrama de actividades. Es decir como procede a fin de ejecutar los proyectos disponibles.

Podemos visualizar en el diagrama izquierdo que se busca recoger toda la información posible, se elabora documentos, se planifica y estudia la manera en que el proyecto será ejecutado.

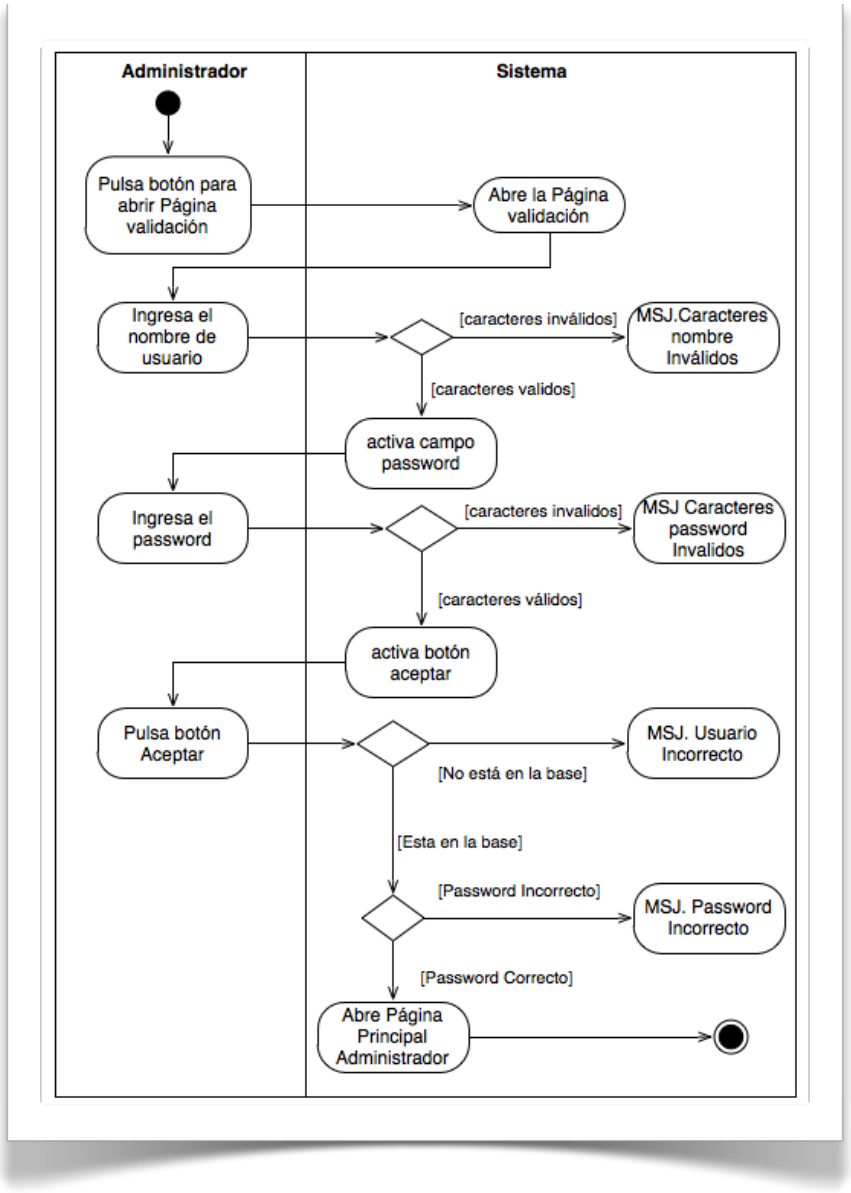
En el gráfico de la derecha vemos que con la información adquirida se busca ejecutar el proyecto.

Sea cual sea la etapa del flujo de actividades el sistema propuesto esta en capacidad de encontrar el candidato (Organización) mas optimo para la pronta ejecución del proyecto.

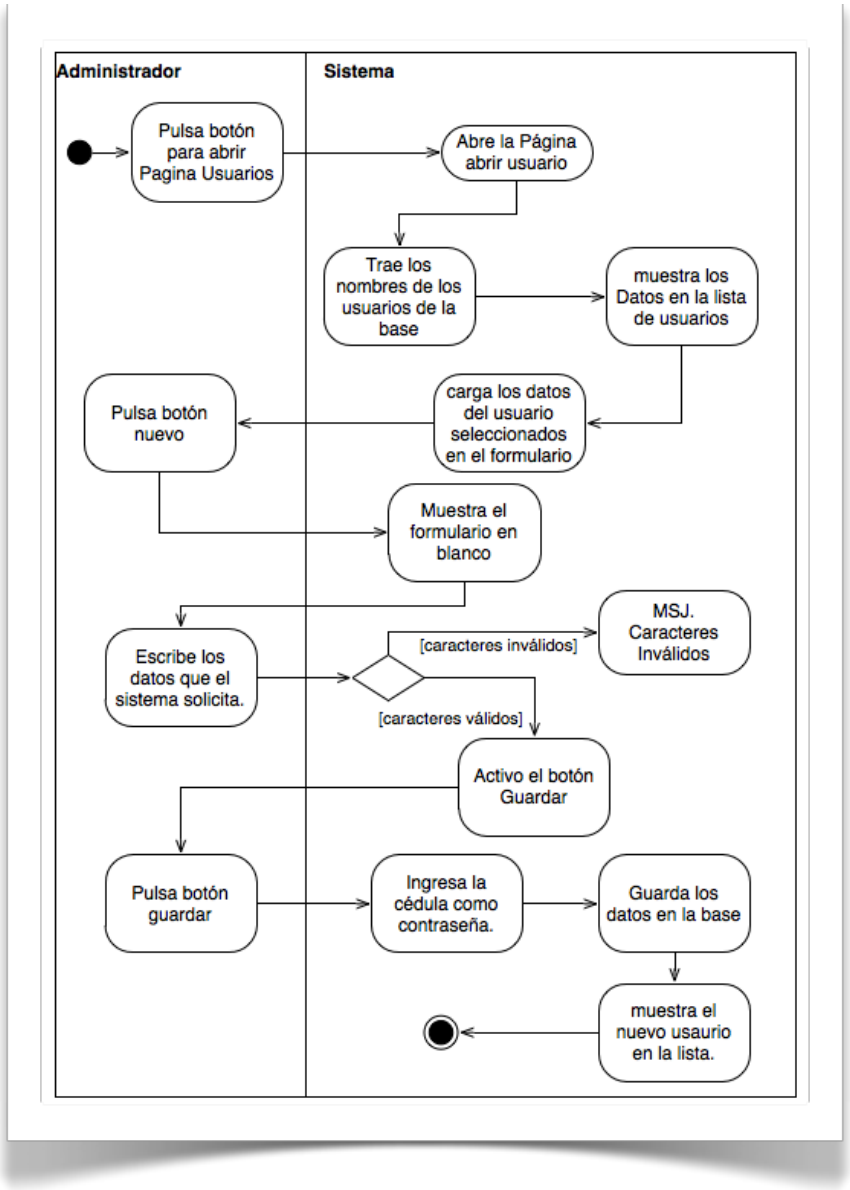
Cabe aclarar que la aplicación se enfoca en el carril de la organización de modo que su objetivo es encontrar quien o quienes pueda desarrollar una parte, o el proyecto completo de ser posible, según sean los alcances de la organización.

Lo referente a proyectos que pueden ser desarrollados por la PUCE esta fuera del alcance de la aplicación ya que esta se enfoca en la búsqueda fuera de la Universidad sin duda es posible aumentar este modulo en una futura actualización.

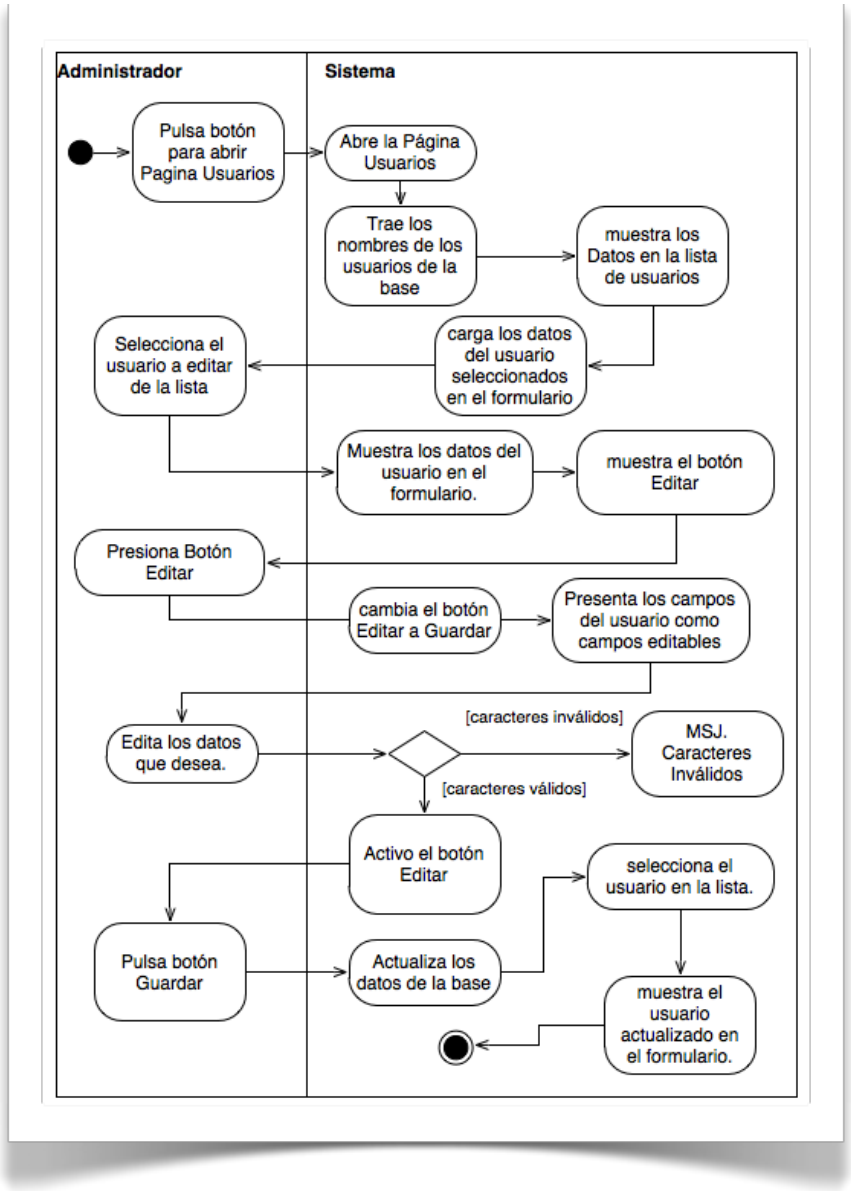
Diagrama de Actividades
Ingreso al Sistema.



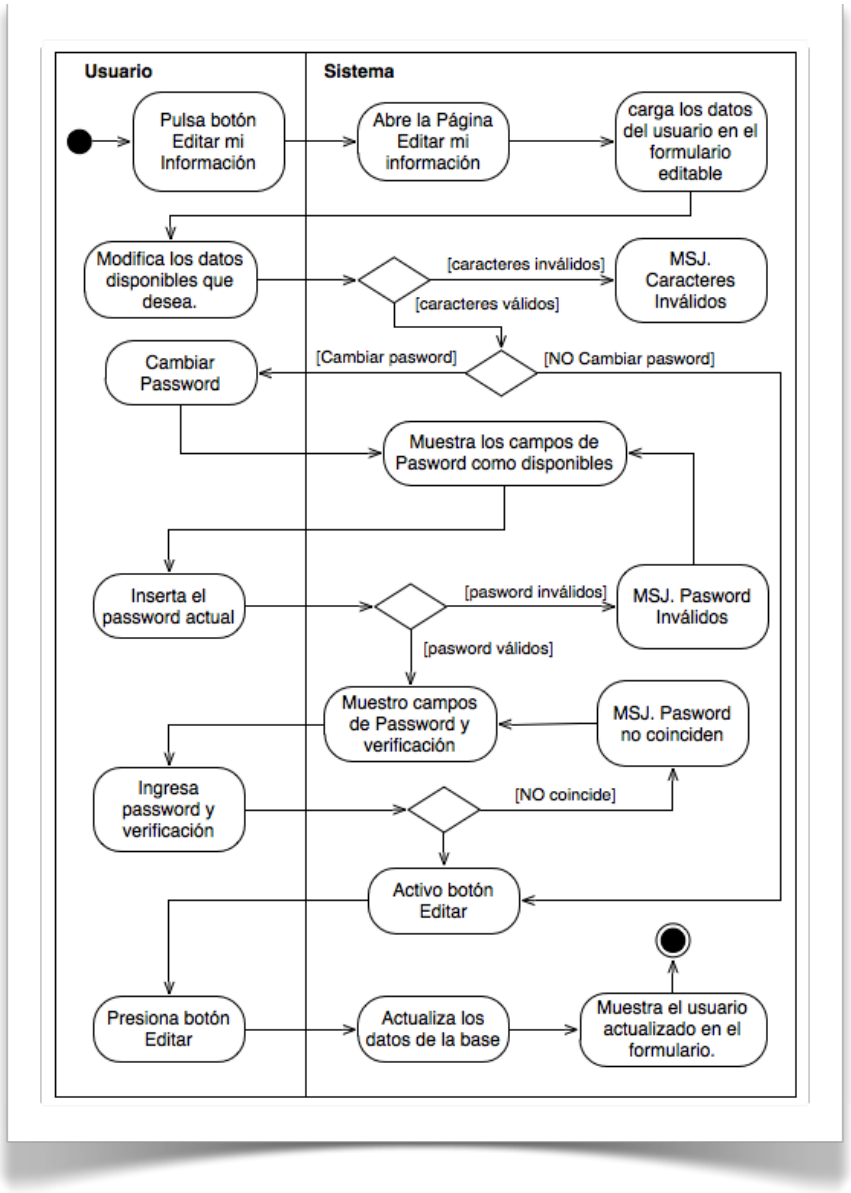
Nuevo Usuario



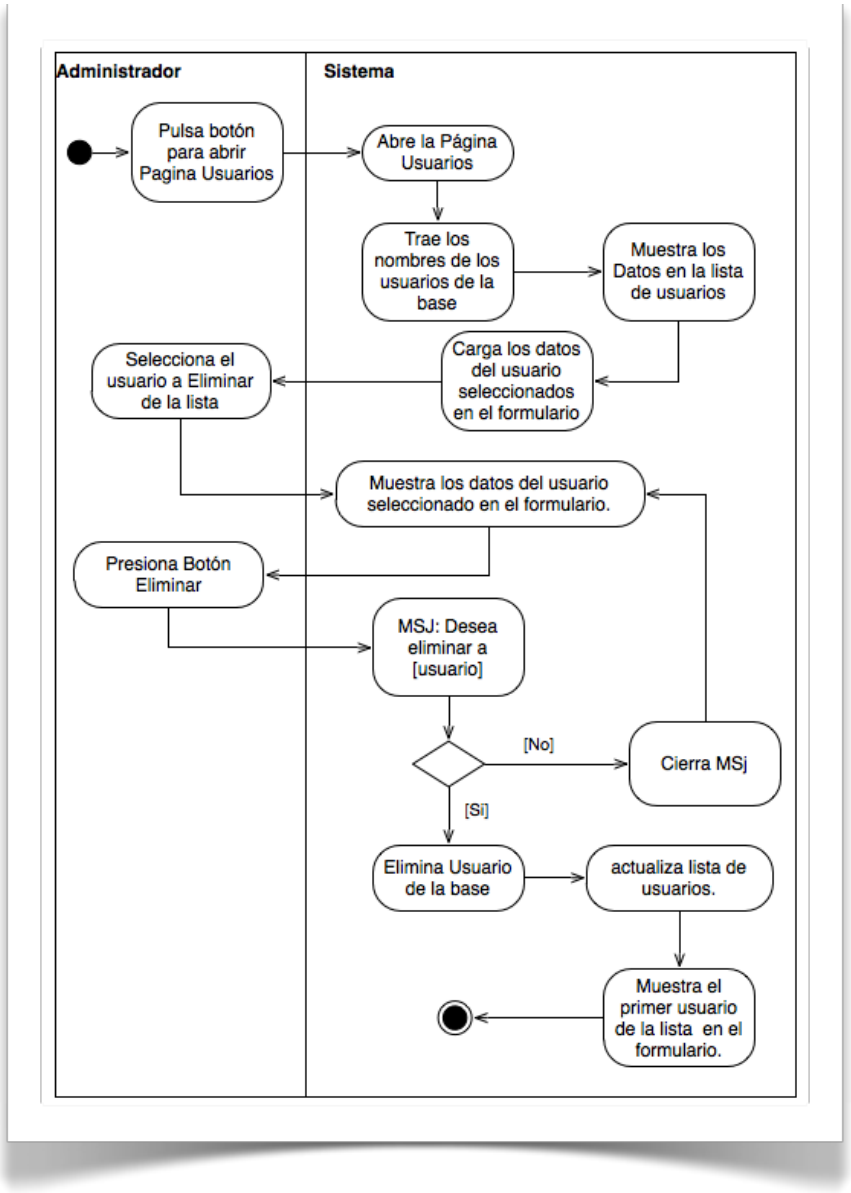
Editar Usuario.



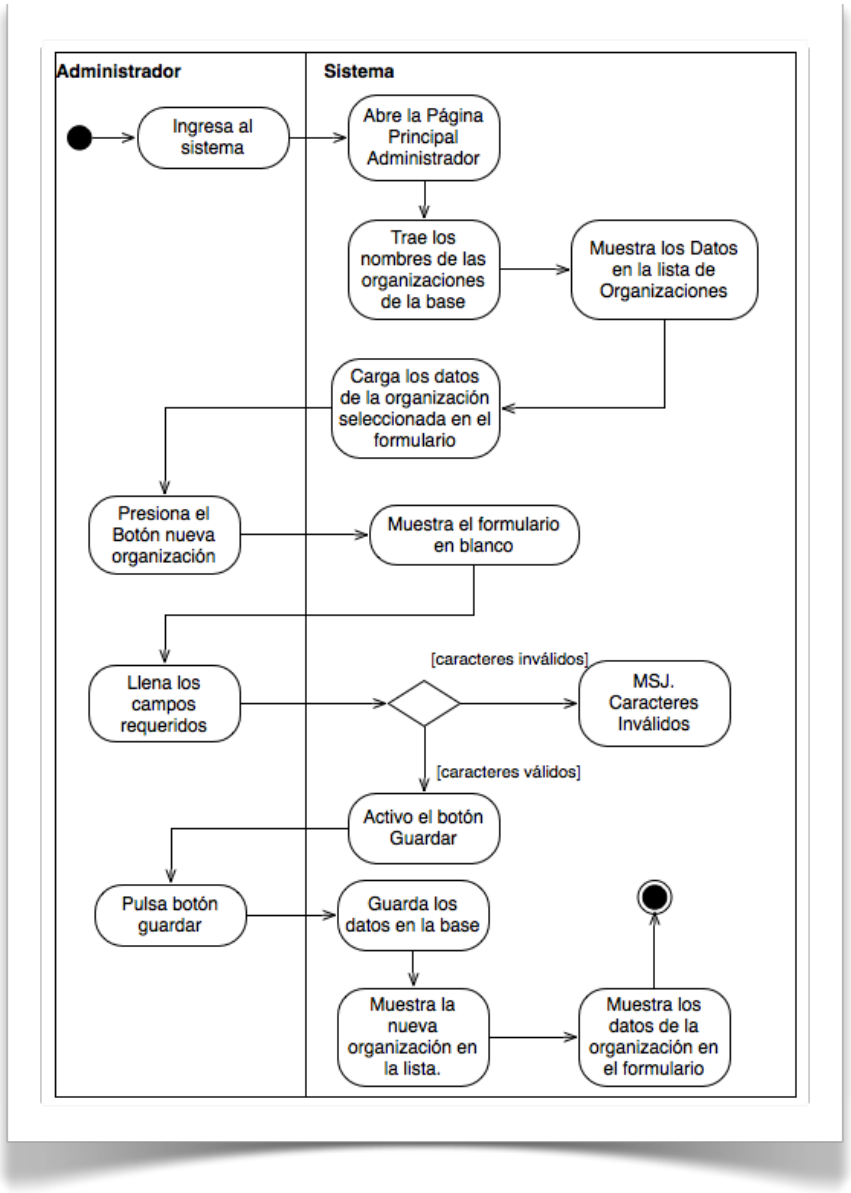
Editar mi Información.



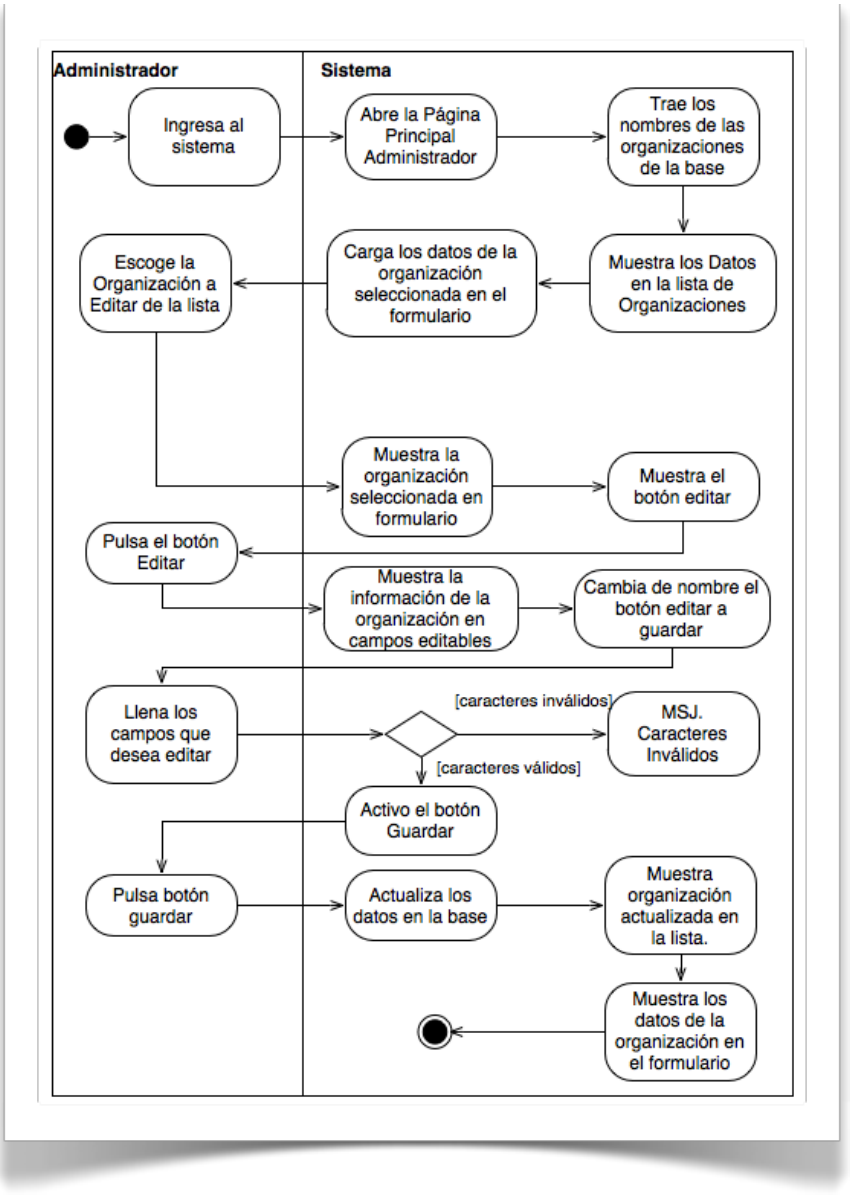
Eliminar Usuario.



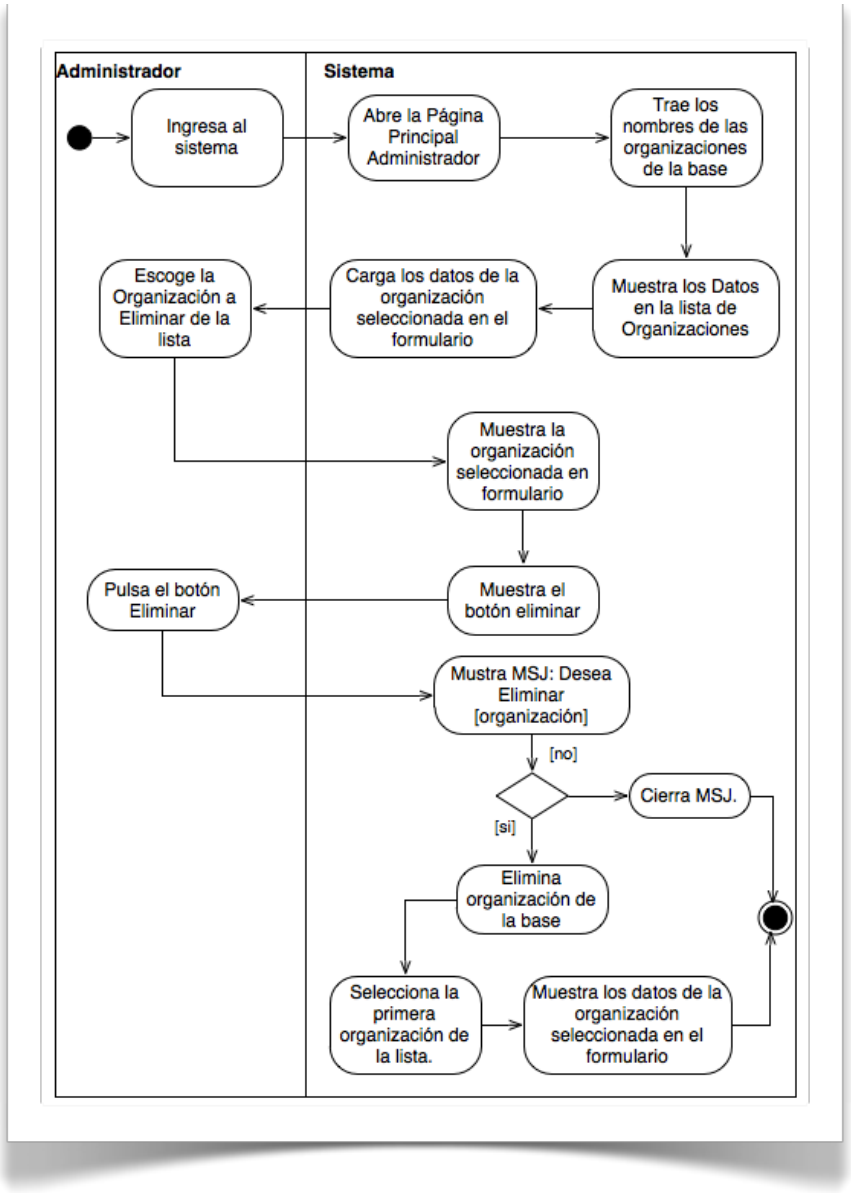
Nueva Organización.



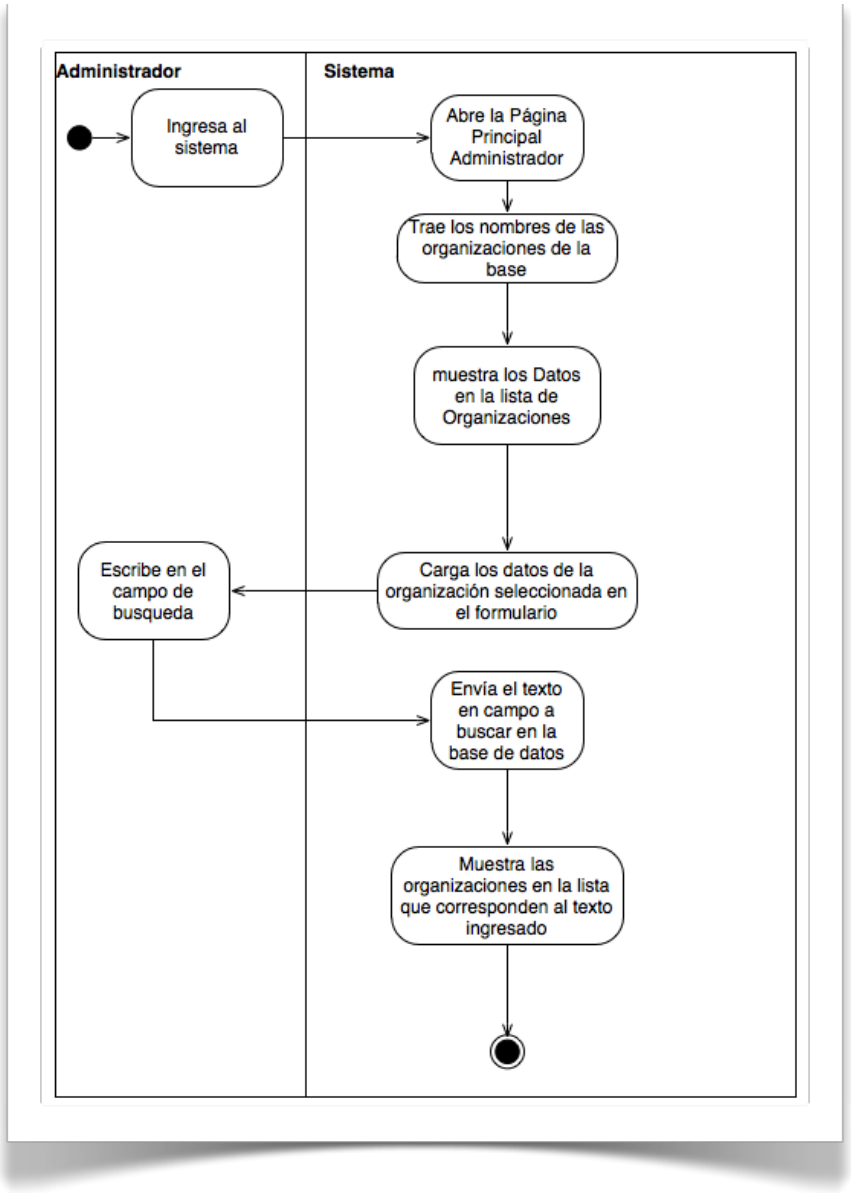
Editar Organización.



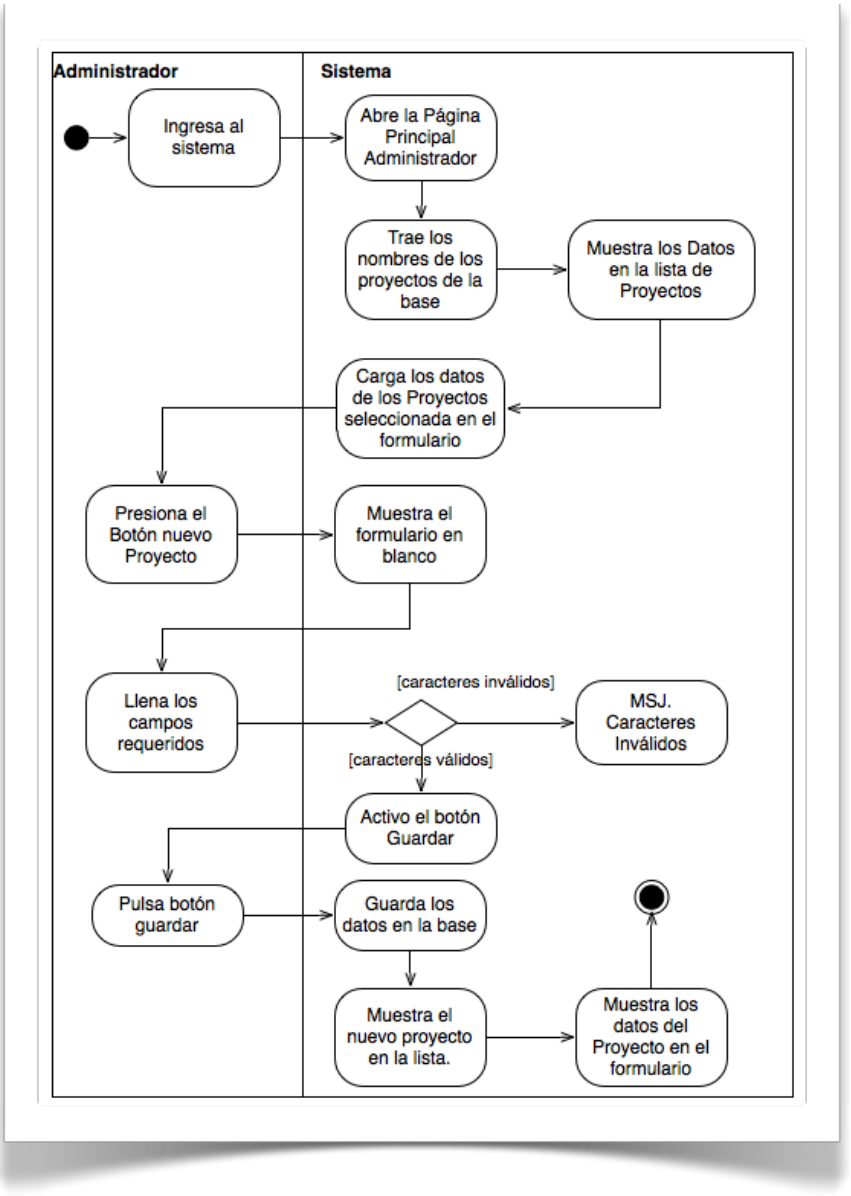
Eliminar Organización.



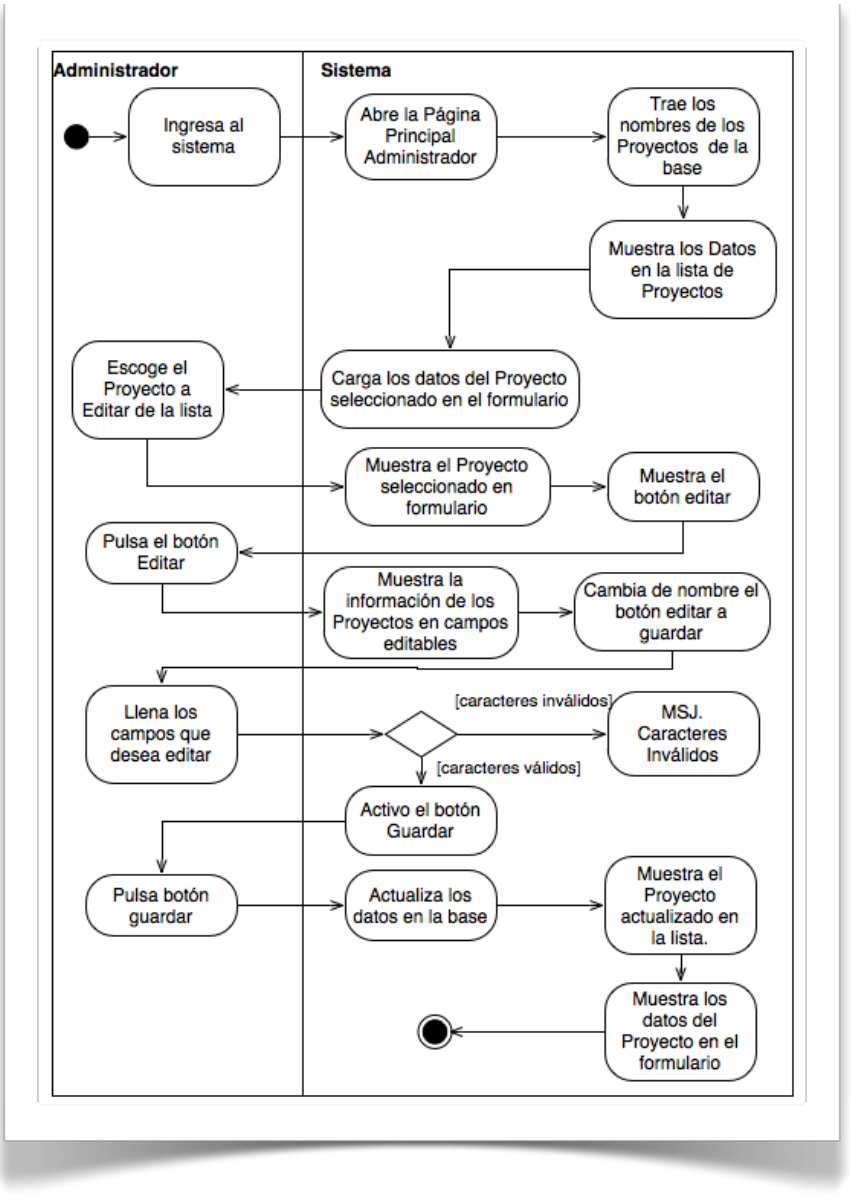
Consultar Organización por Nombre.



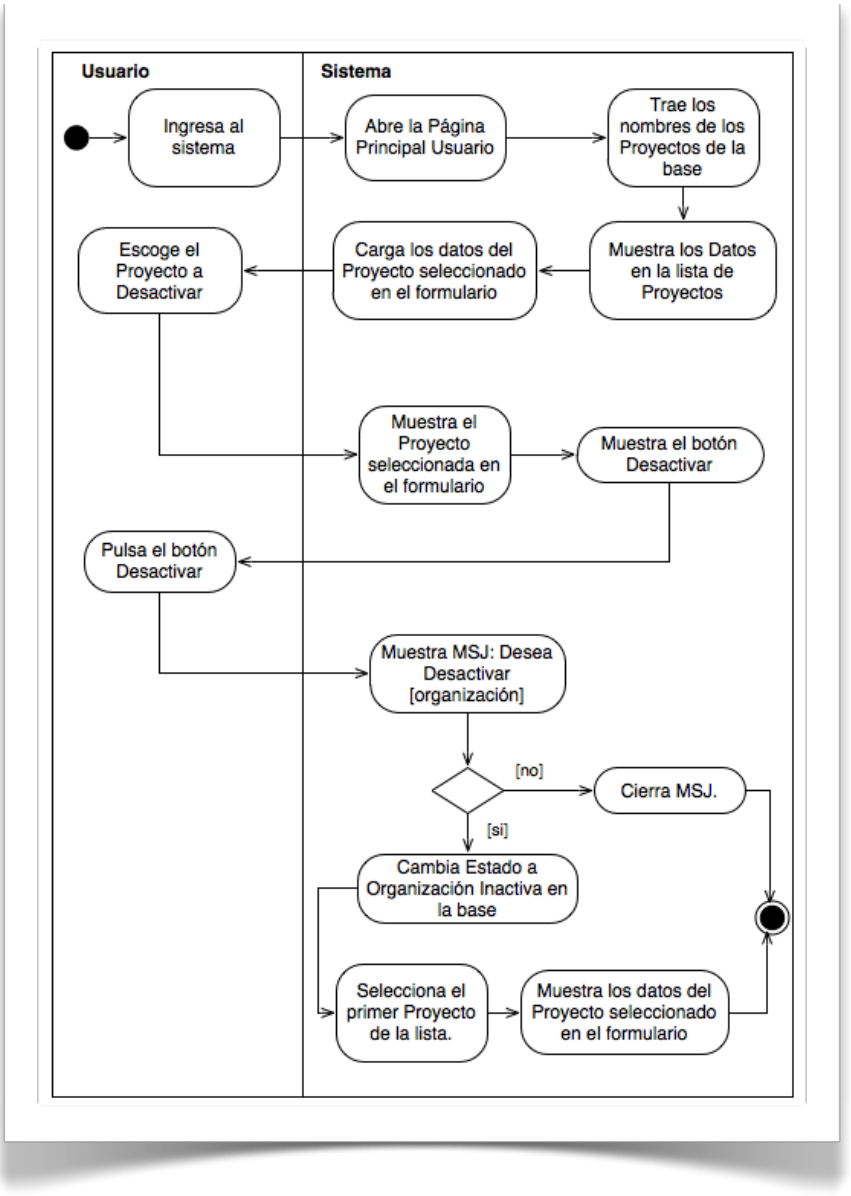
Nuevo Proyecto.



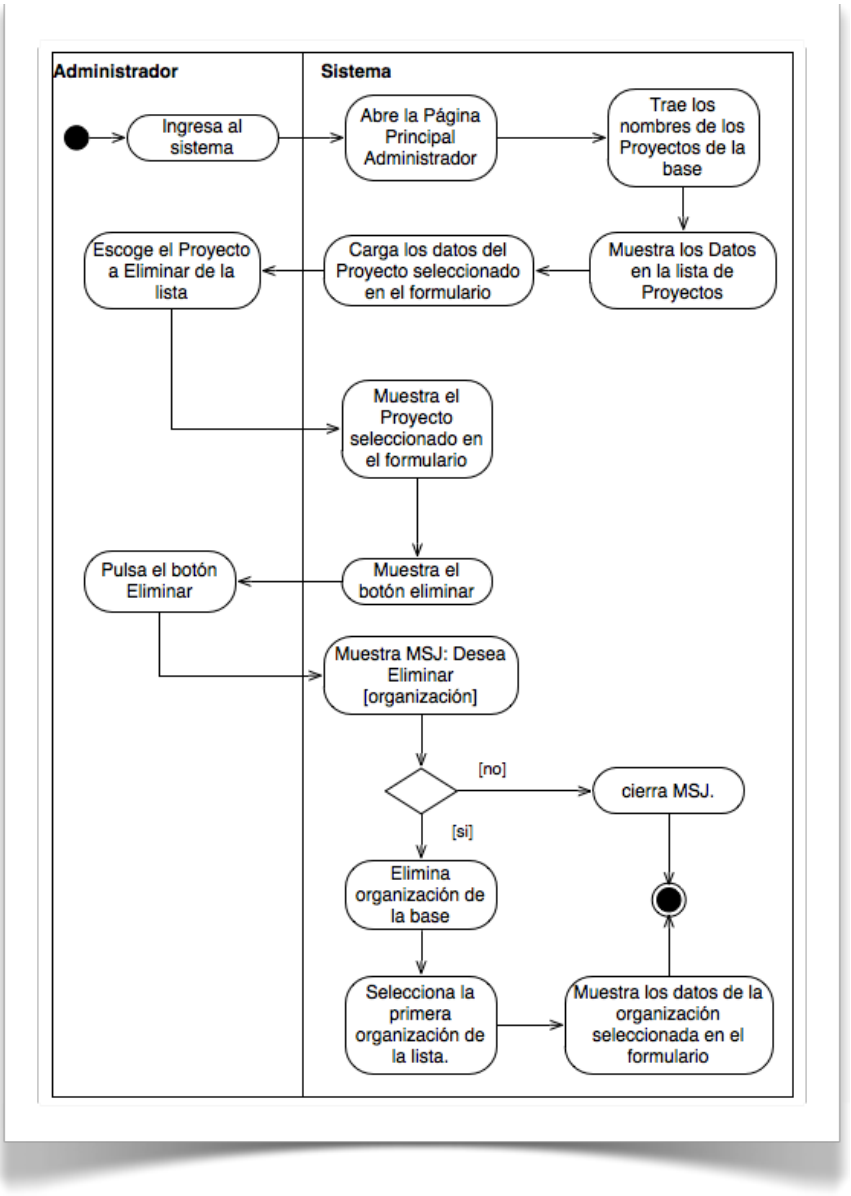
Editar Proyecto.



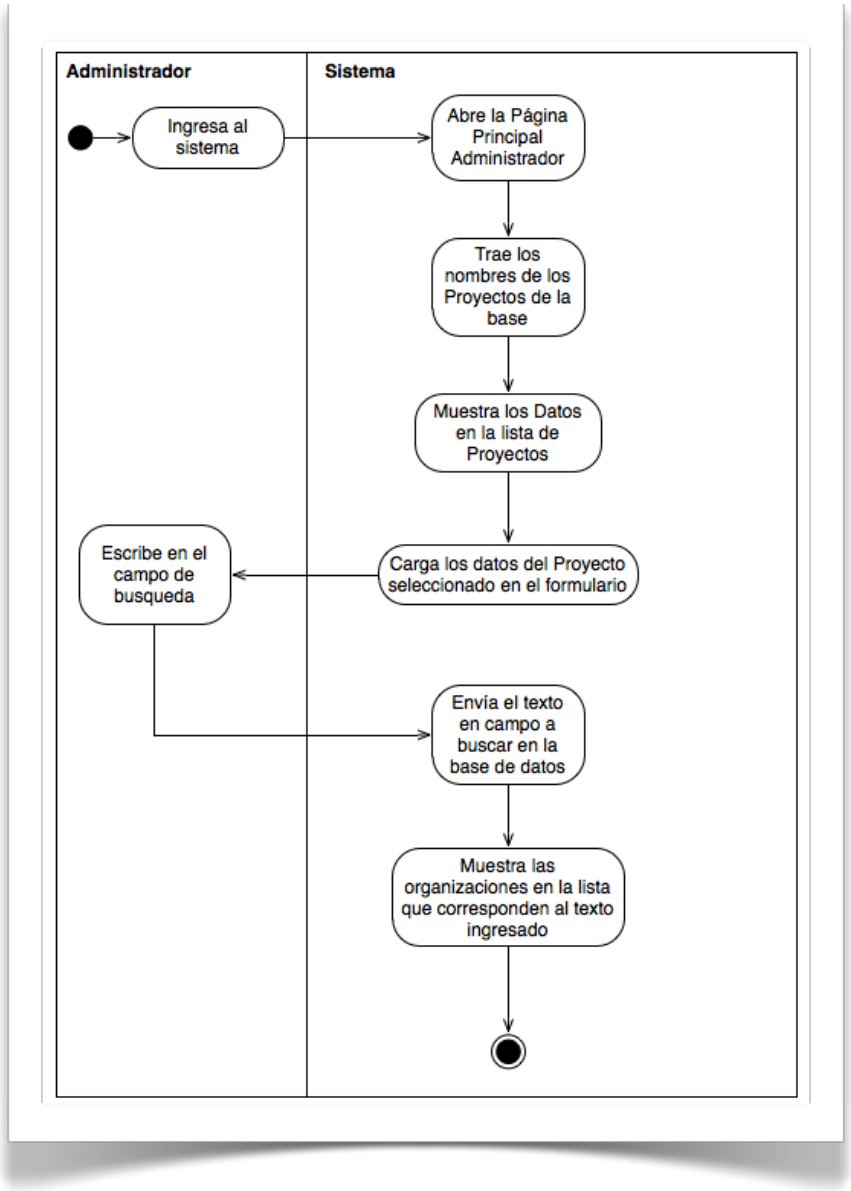
Desactivar Proyecto.



Eliminar Proyecto.



Consulta Proyecto por Nombre.



Salir del Sistema.

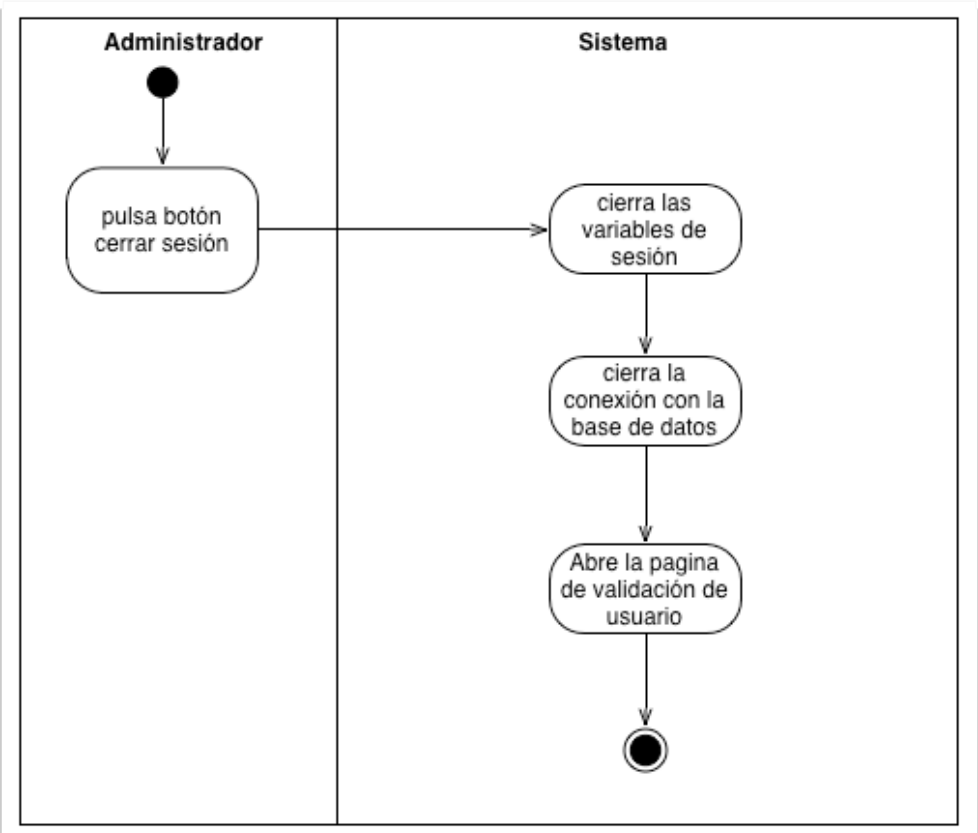


Diagrama de Clases.
Nivel Conceptual

El diagrama de clases a nivel conceptual describe la estructura del sistema de modo que se visualiza las clases existentes sin métodos ni atributos

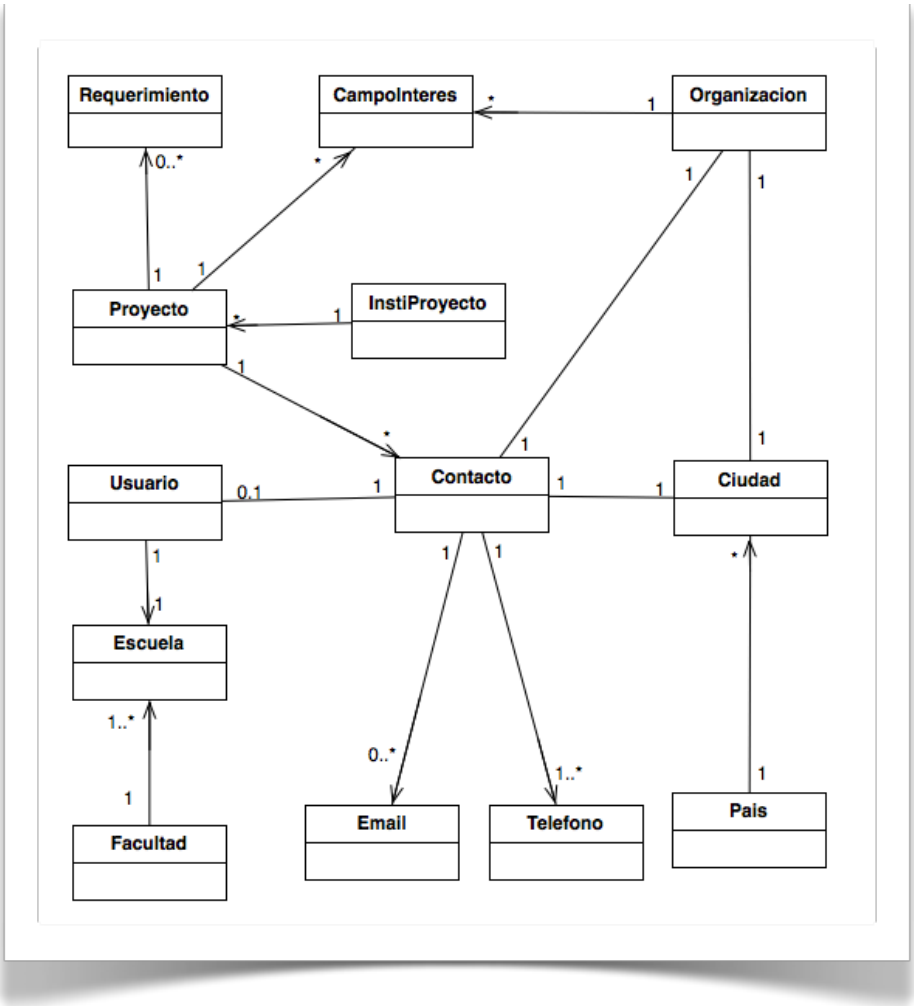
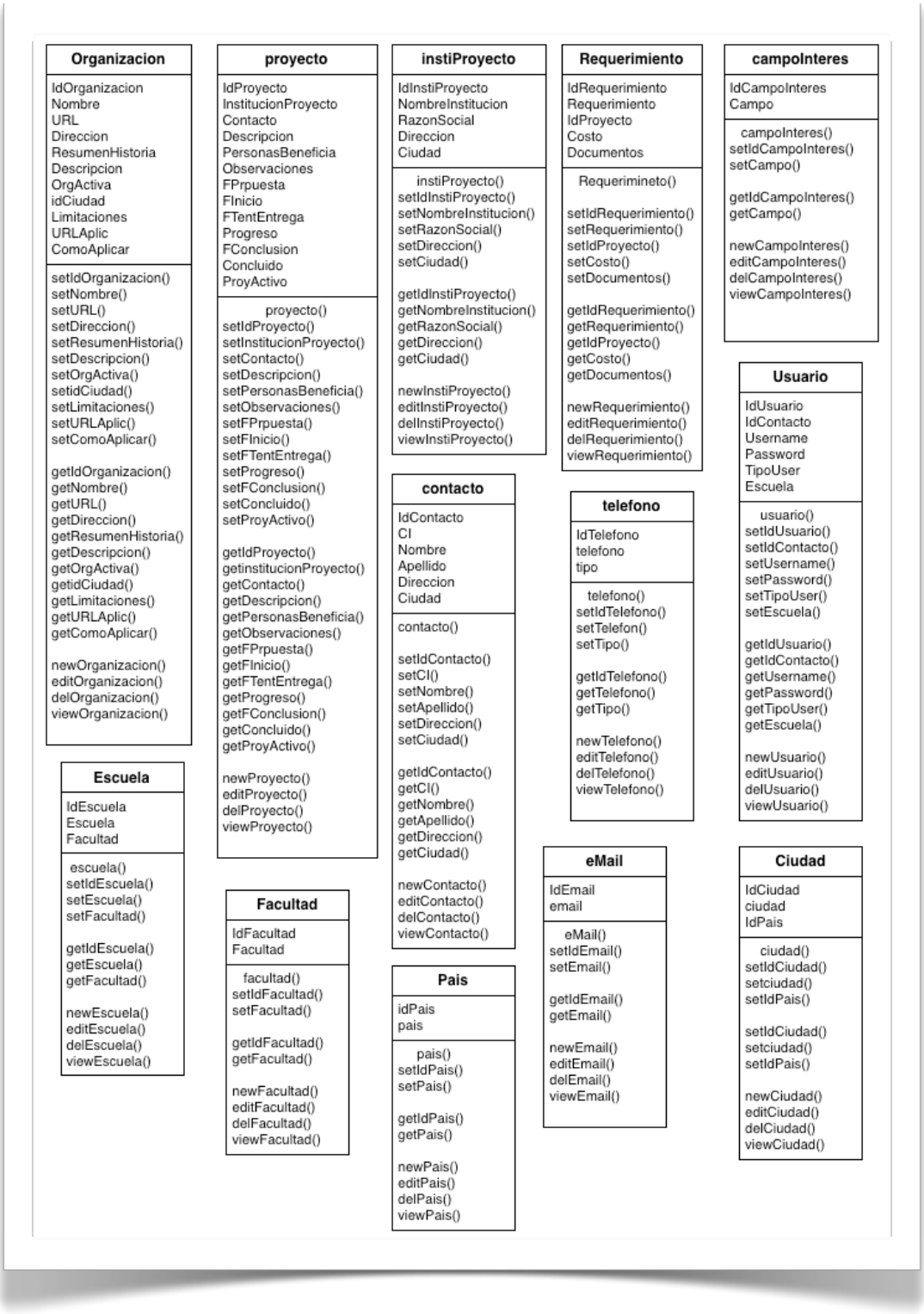


Diagrama de Clases.
Nivel Detalle

El diagrama de clases a nivel a detalle visualiza las clases con sus respectivos métodos y atributos.



Patrón Arquitectónico MVC

El MVC (Modelo Vista Controlador) es un patrón de diseño de software que se caracteriza por separar la lógica del negocio de la interfaz del usuario evitando mezclar código de programación con diseño de la interface gráfica, facilitando la actualización y mantenimiento de la aplicación.

El MVC divide la aplicación en tres capas o niveles.

Modelo

Vista

Controlador

Modelo: Es la capa que se encarga de manejar la lógica del negocio. Es la que accede a los datos, ejecuta operaciones entre clases (validación de datos, cálculos y operaciones etc.)

Vista: Es la capa que se encarga de mostrar los datos enviados por el modelo de una manera comprensible para el usuario, la vista será manipulada por los Diseñadores dotando de una manera fácil de leer la información solicitada al modelo.

Controlador: Es el encargado de comunicar la vista con el modelo.

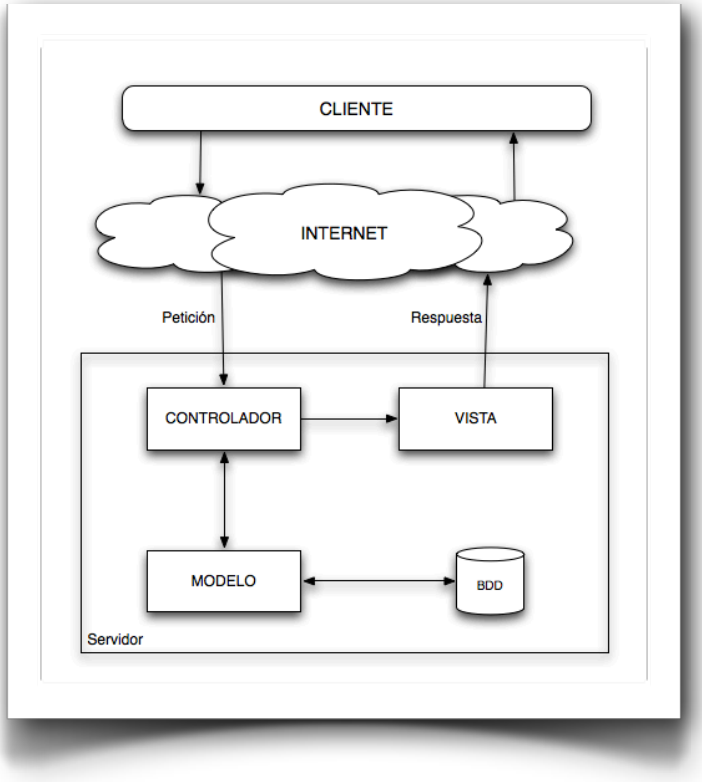


Figura 5: Ilustración del Modelo Vista Controlador

Elaborado por: Darshan Zamora

Diagrama de Secuencia.

UML con los diagramas de secuencia nos permite expandir nuestro campo de visión y nos muestra cómo un objeto interactúa con otros objetos. En estos diagramas se añade una nueva dimensión, el tiempo. La idea clave es que los objetos interactúen en una secuencia especifica y esta secuencia toma un tiempo para ir de inicio a fin.

Los diagramas de secuencia se representan por:

- Objetos.
- Mensajes.
- Tiempo.

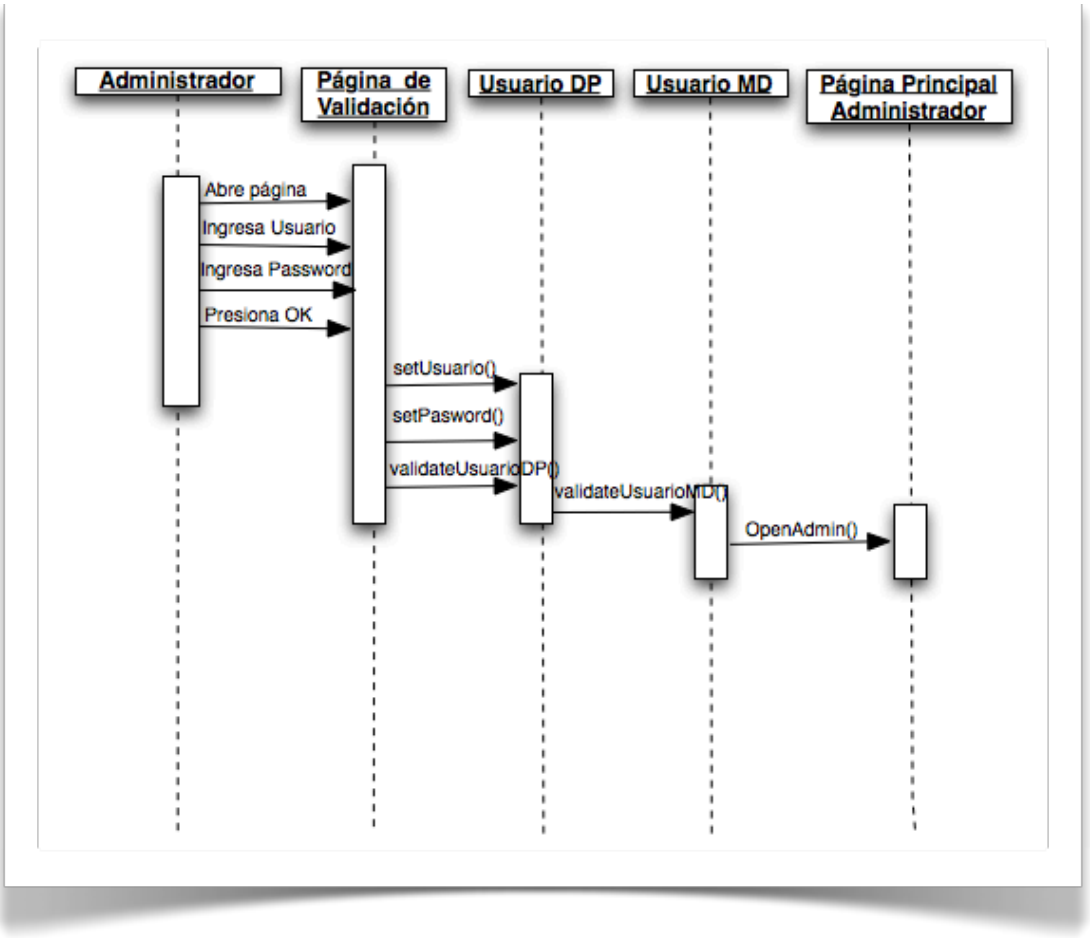
Objetos. Representados por rectángulos con el nombre del objeto subrayado, ubicado en la parte superior del diagrama distribuido horizontalmente de izquierda a derecha, de manera que el diagrama se simplifique. Debajo de cada objeto está una línea punteada perpendicular al rectángulo llamada línea de vida o línea de tiempo.

Mensajes. Son representados por una flecha que une los objetos por medio de la línea de vida o a su propia línea de vida. Representa los mensajes enviados entre objetos. La forma de la punta de flecha muestra qué tipo de mensaje está enviando; si es sincrónico o asincrónico. El mensaje sincrónico quiere decir que el mensaje detiene el sistema hasta que termine la llamada. Este mensaje es representado por una flecha con la punta llena y el cuerpo por una línea continua y el mensaje asíncrono es el que no importa si terminó el mensaje. Se representan con la cabeza de la flecha abierta y linea entrecortada.

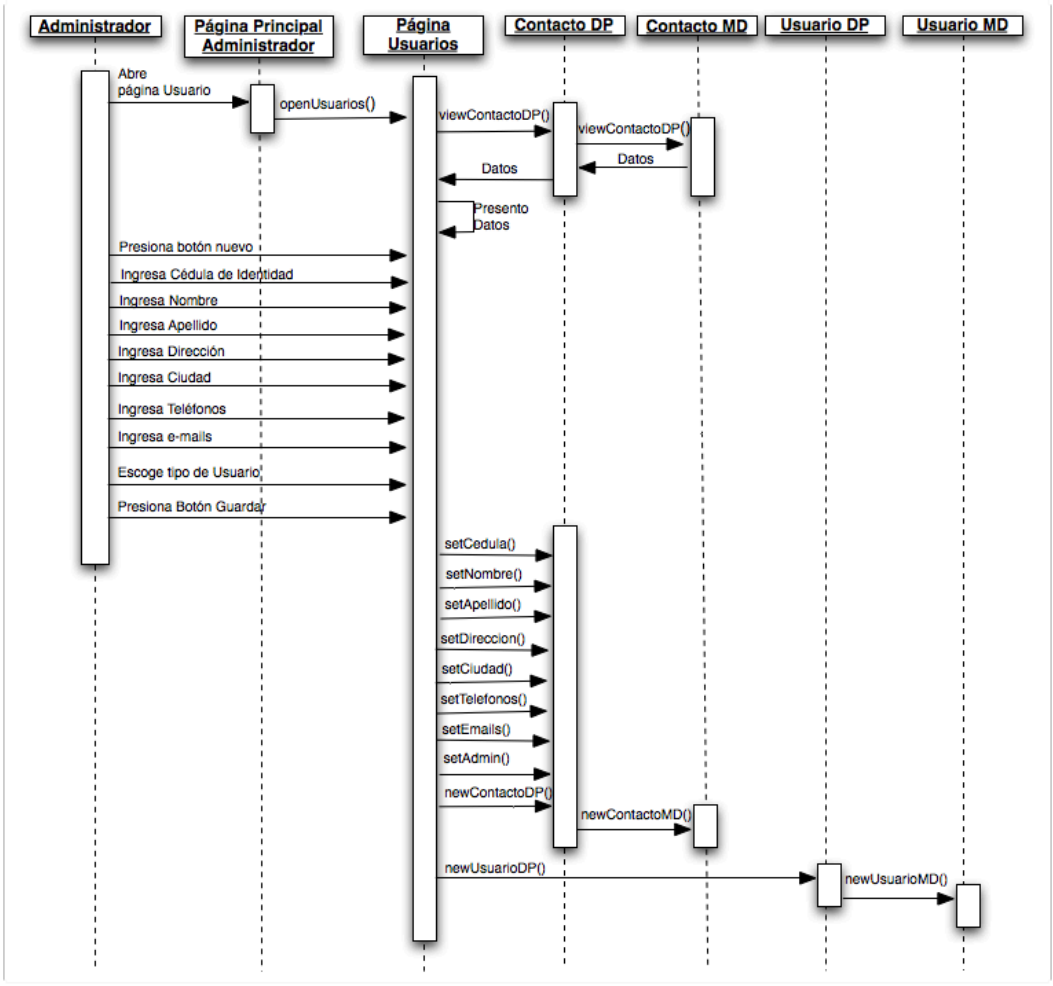
Tiempo. Es representado como la progresión vertical desde la parte superior del diagrama a la parte inferior.

A continuación se presentan los modelos asociados a los casos de uso expuestos anteriormente, complementando de este modo la comprensión de la funcionalidad del sistema haciendo de este un diseño integral, con el fin de hacer la actualización y el mantenimiento más sencillos.

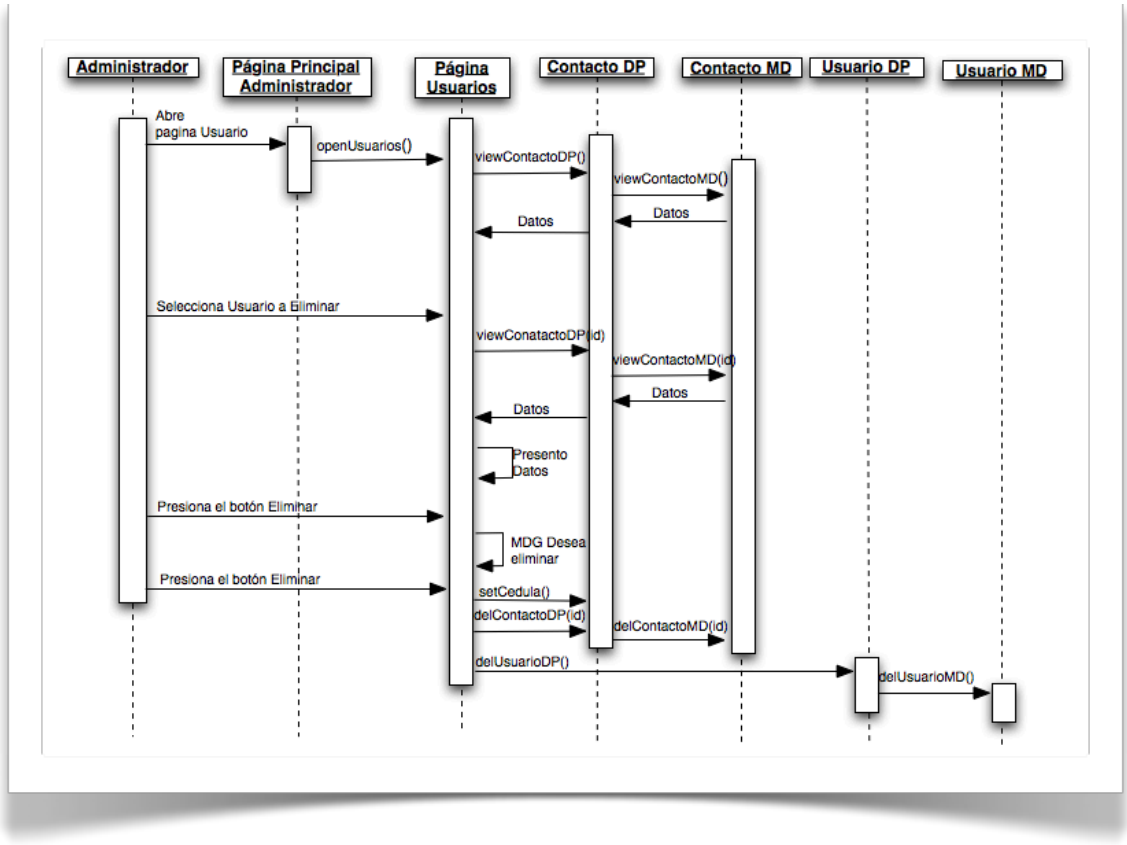
Diagrama de Secuencia.
F1 Ingreso al Sistema.



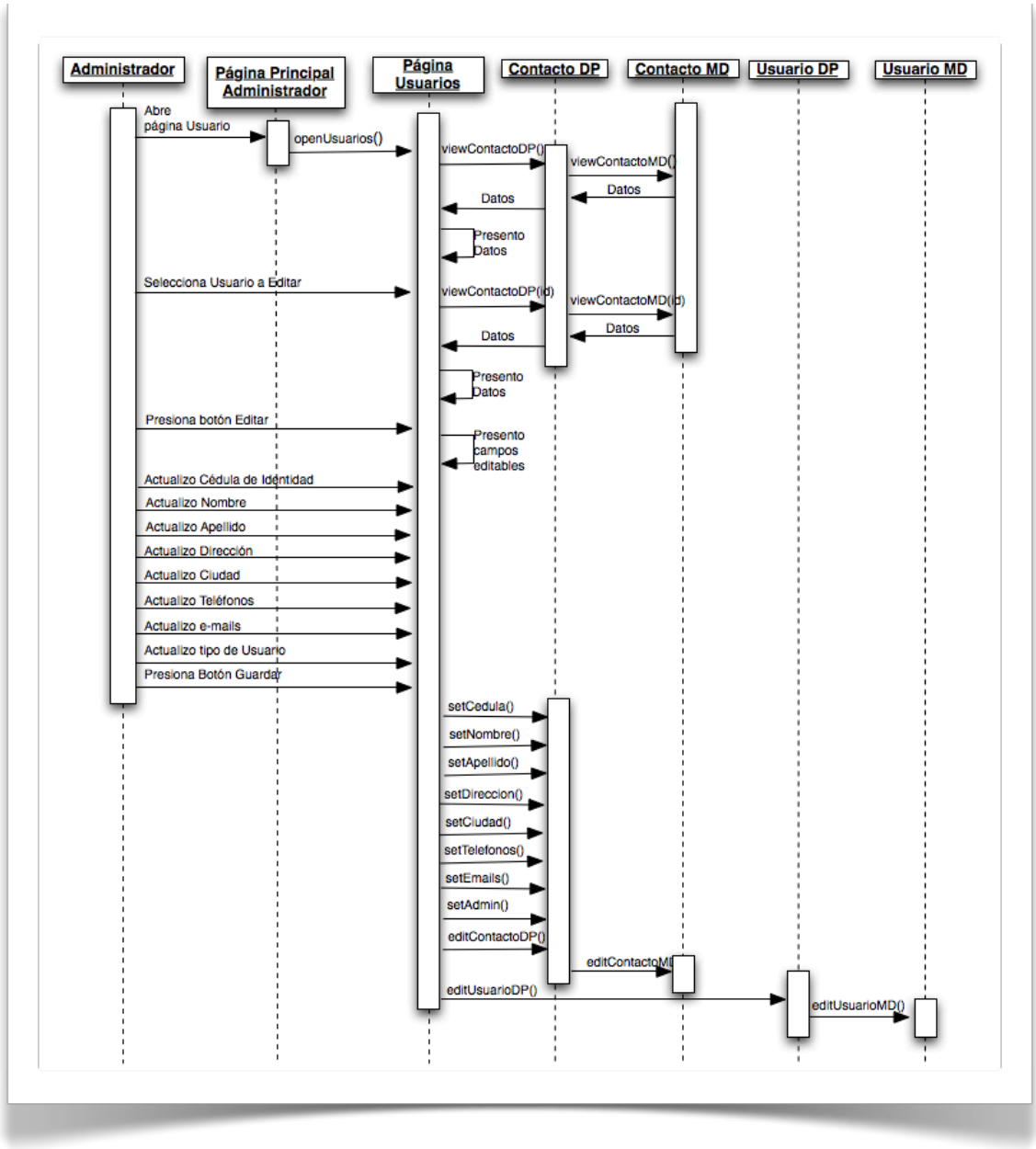
F 2.2 Nuevo Usuario.



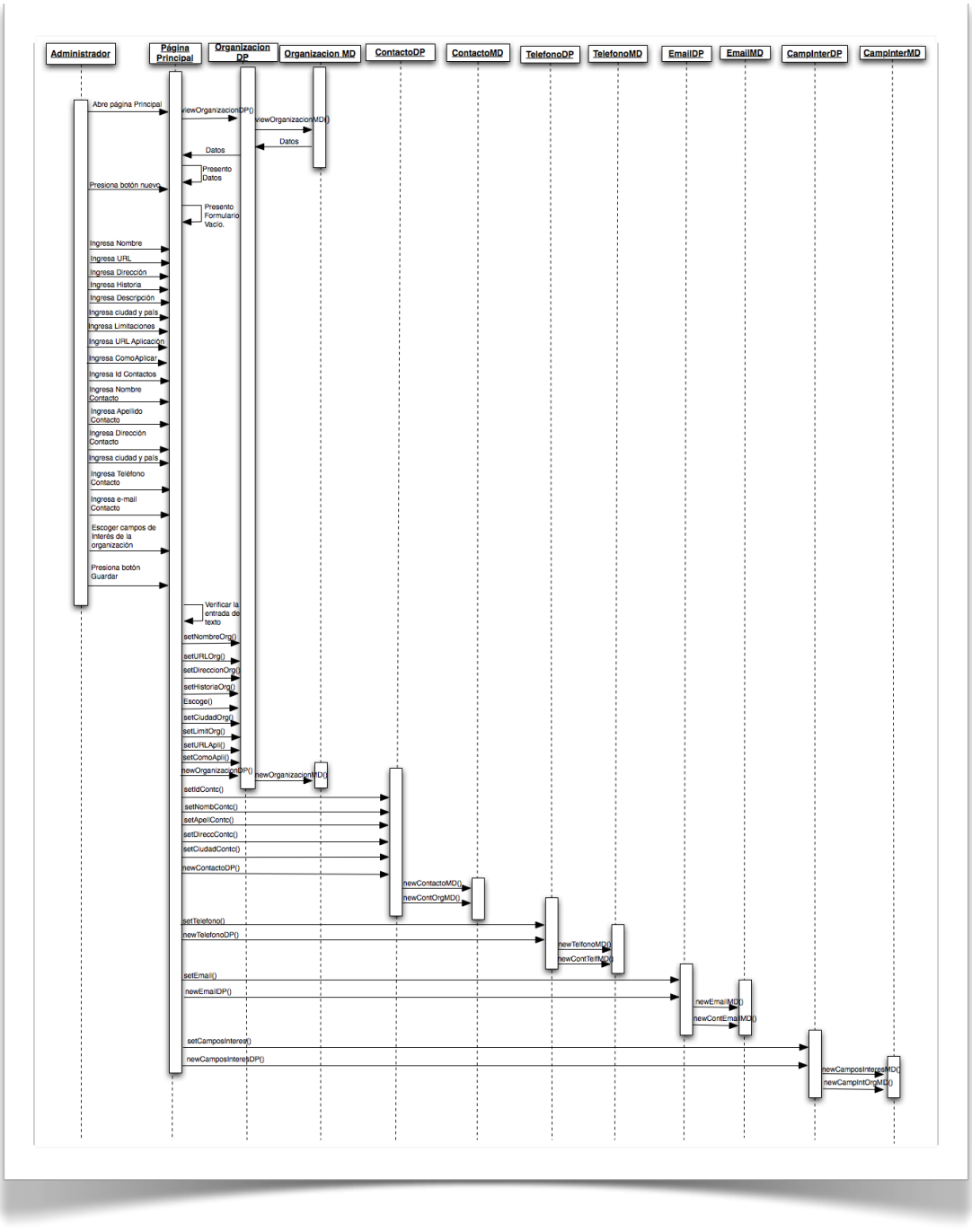
F 2.3 Eliminar Usuario.



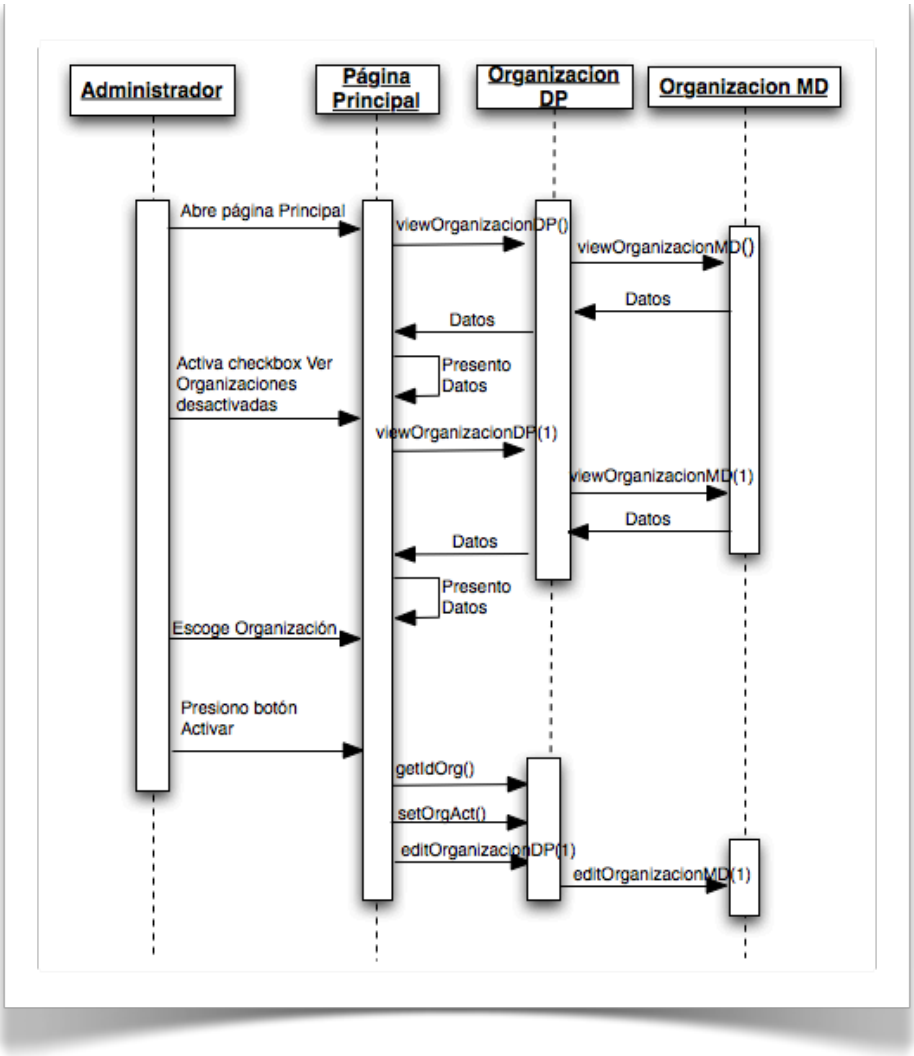
F 2.4 Editar Usuario.



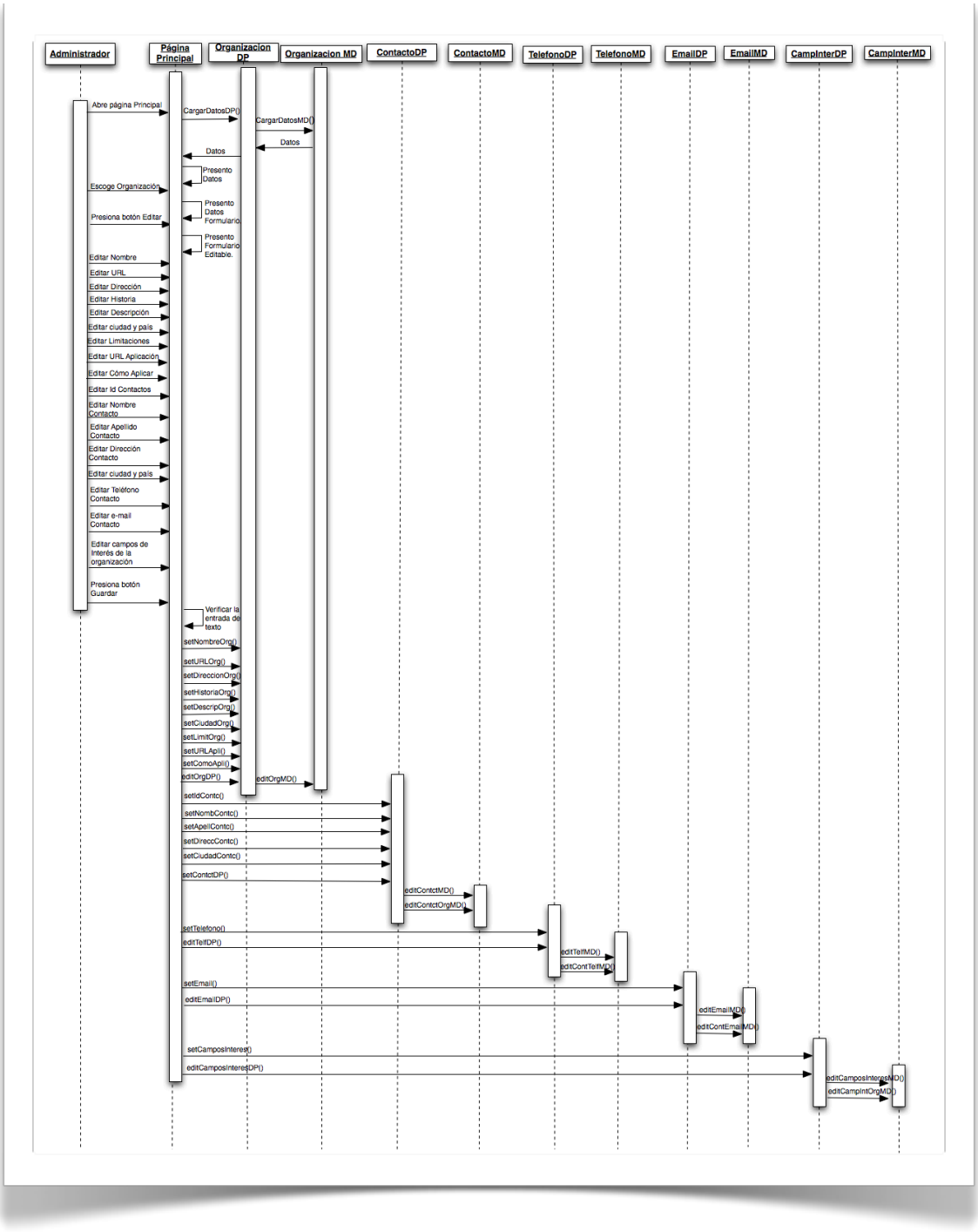
F 3.2 Insertar Organización.



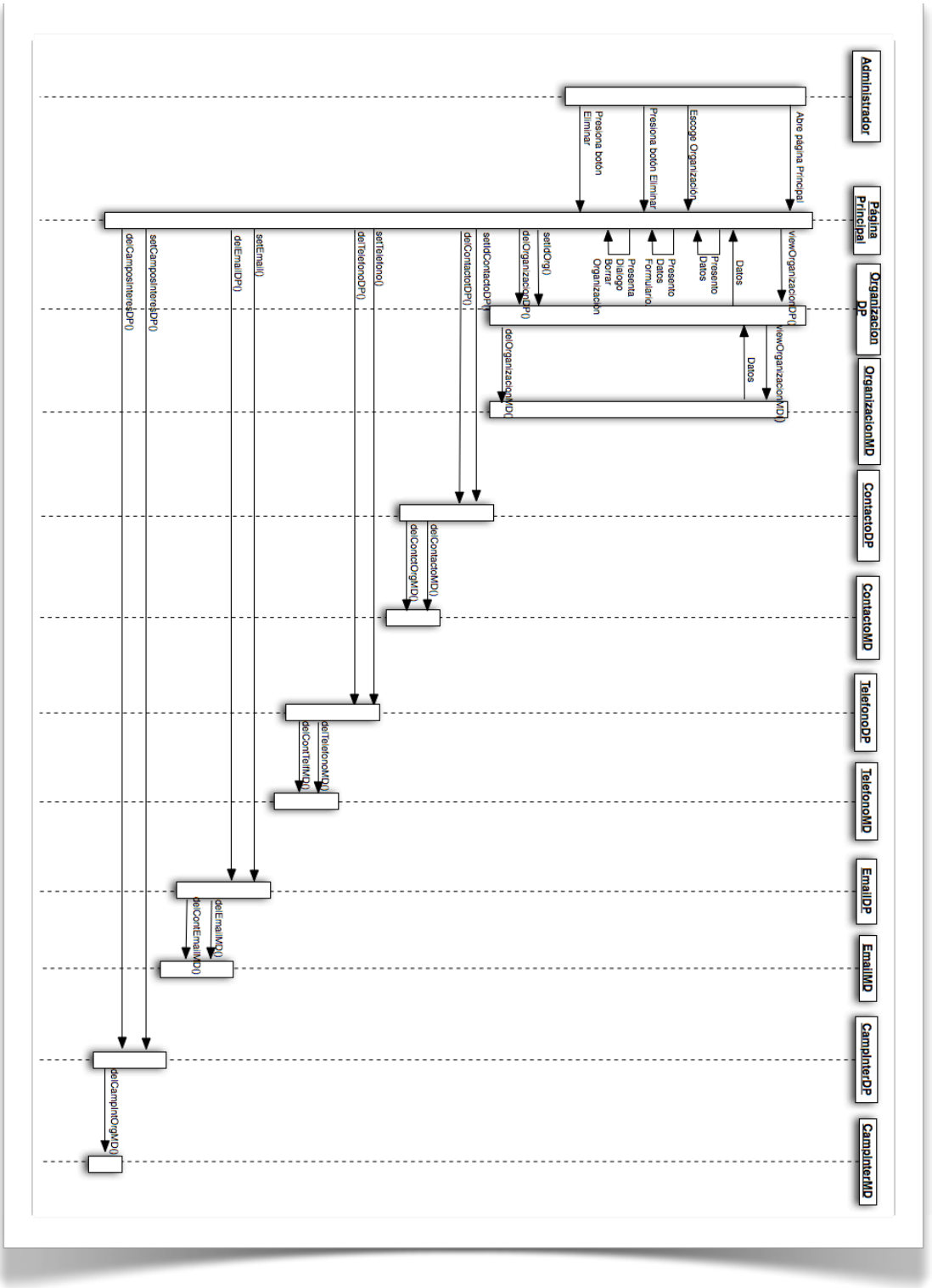
F 3.3 Activar Organización.



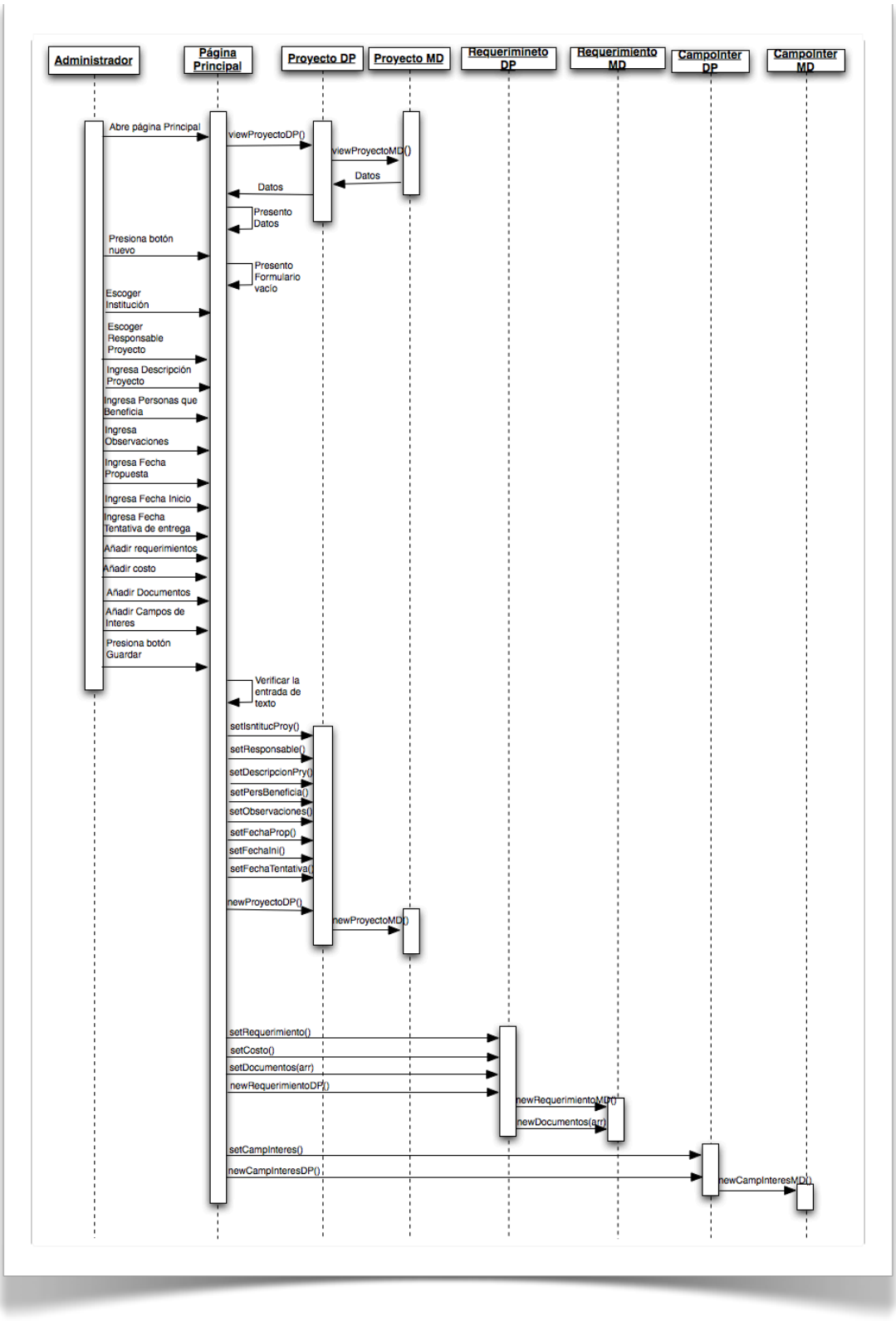
F 3.3 Editar Organización.



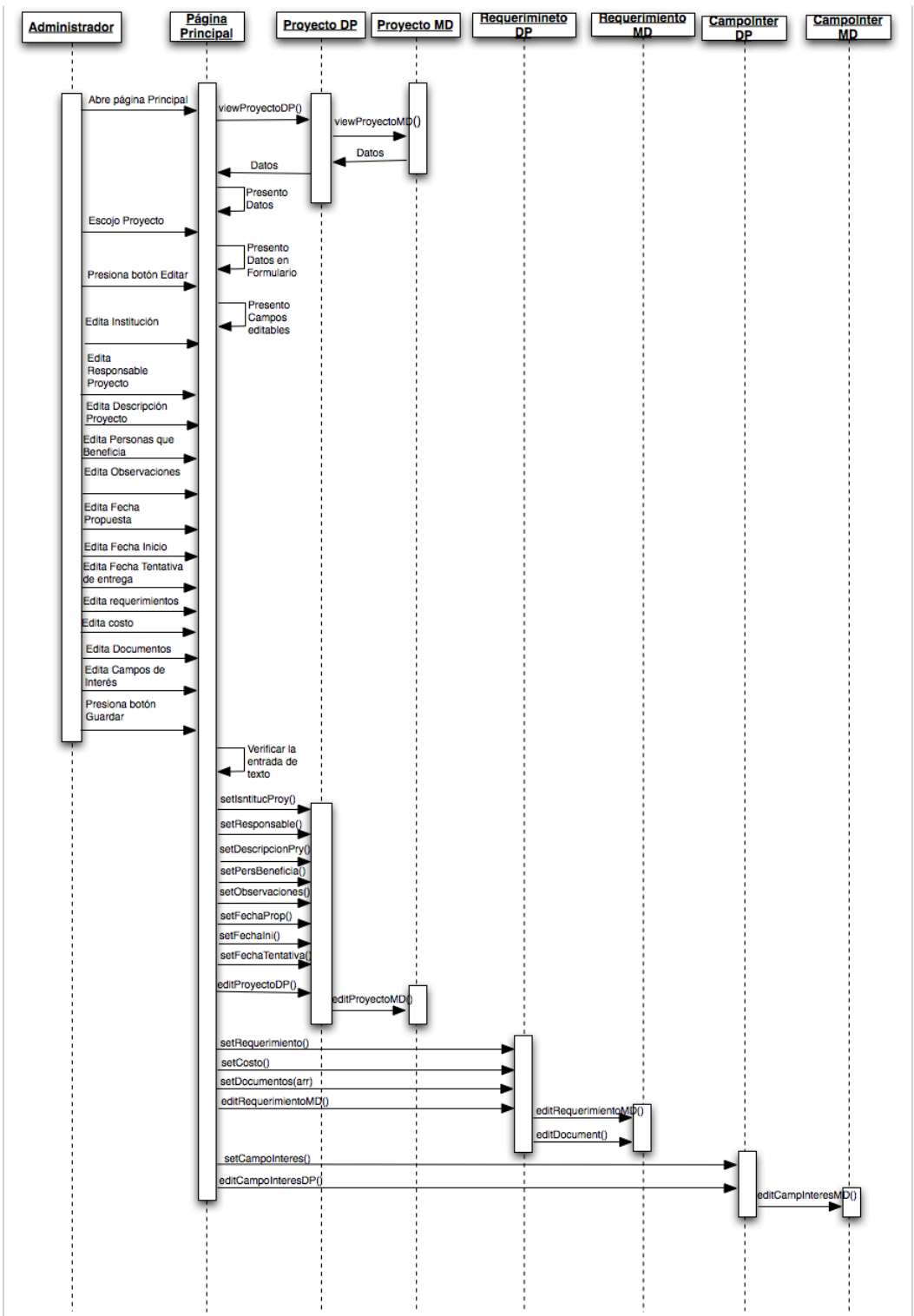
F 3.4 Eliminar Organización.



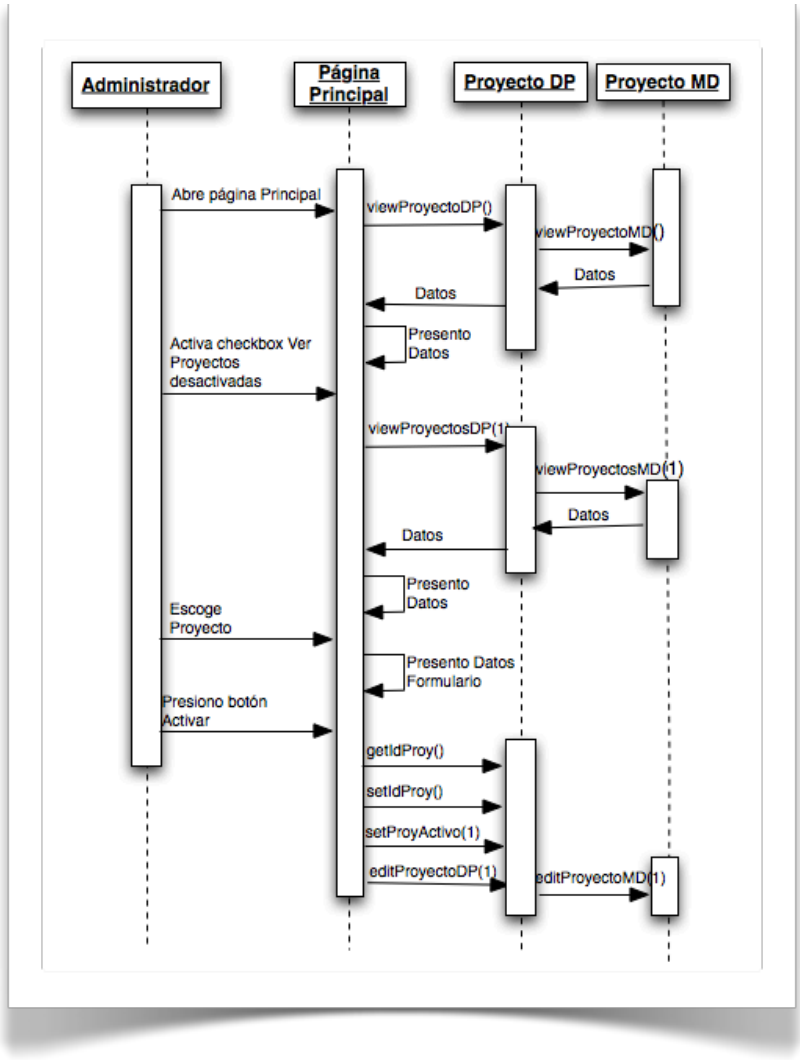
F 4.2 Nuevo Proyecto.



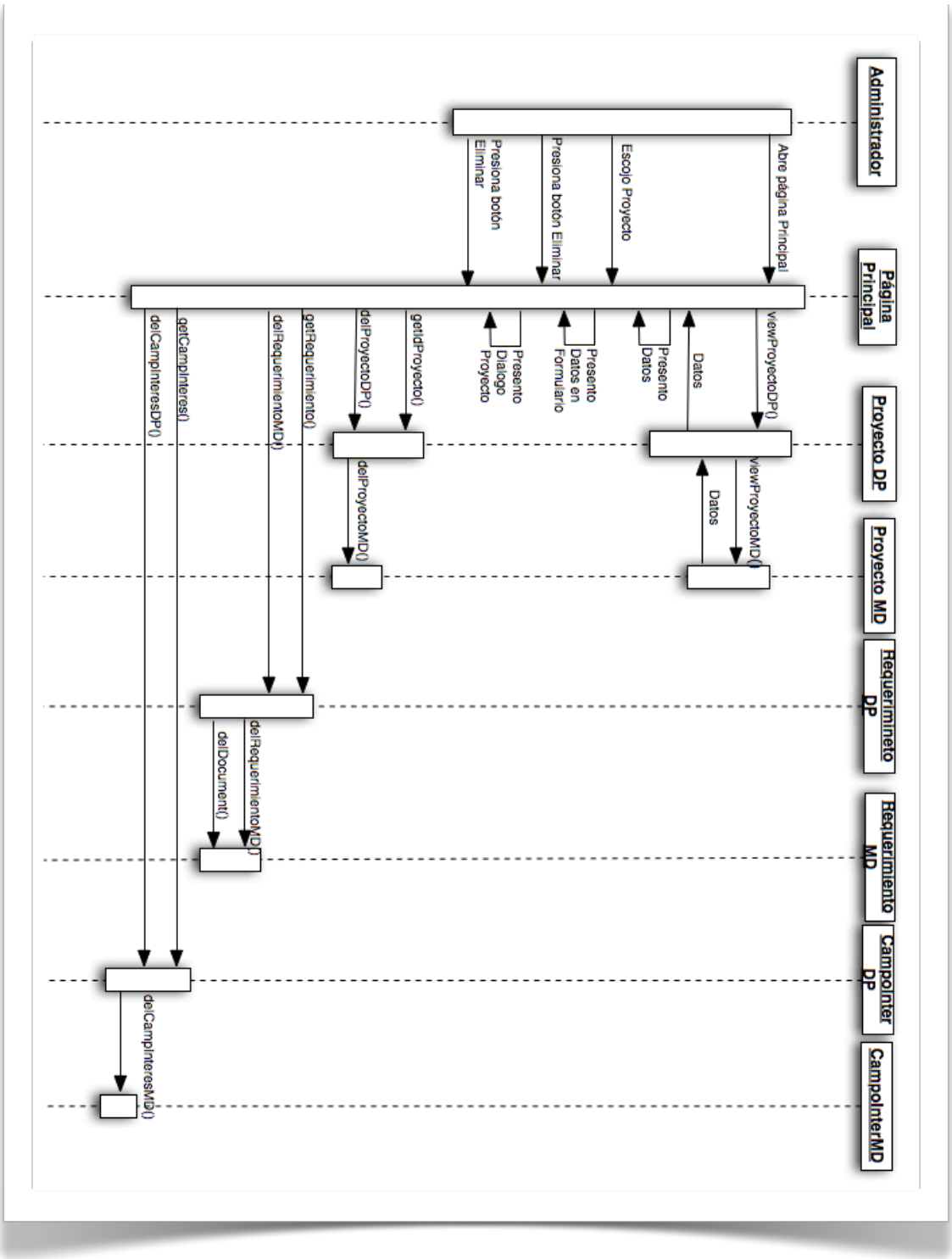
F 4.3 Editar Proyecto.



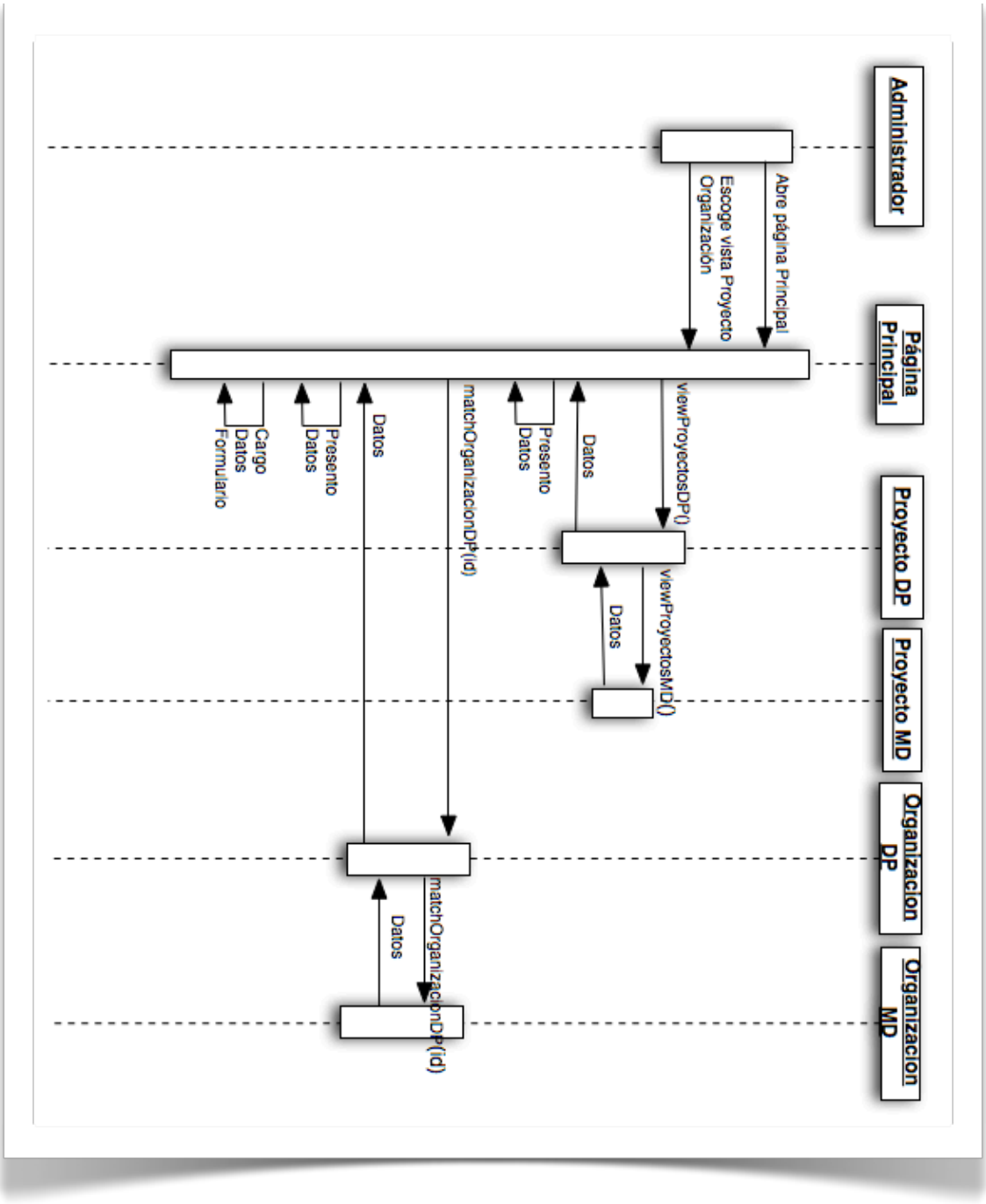
F 4.3 Activar Proyecto.



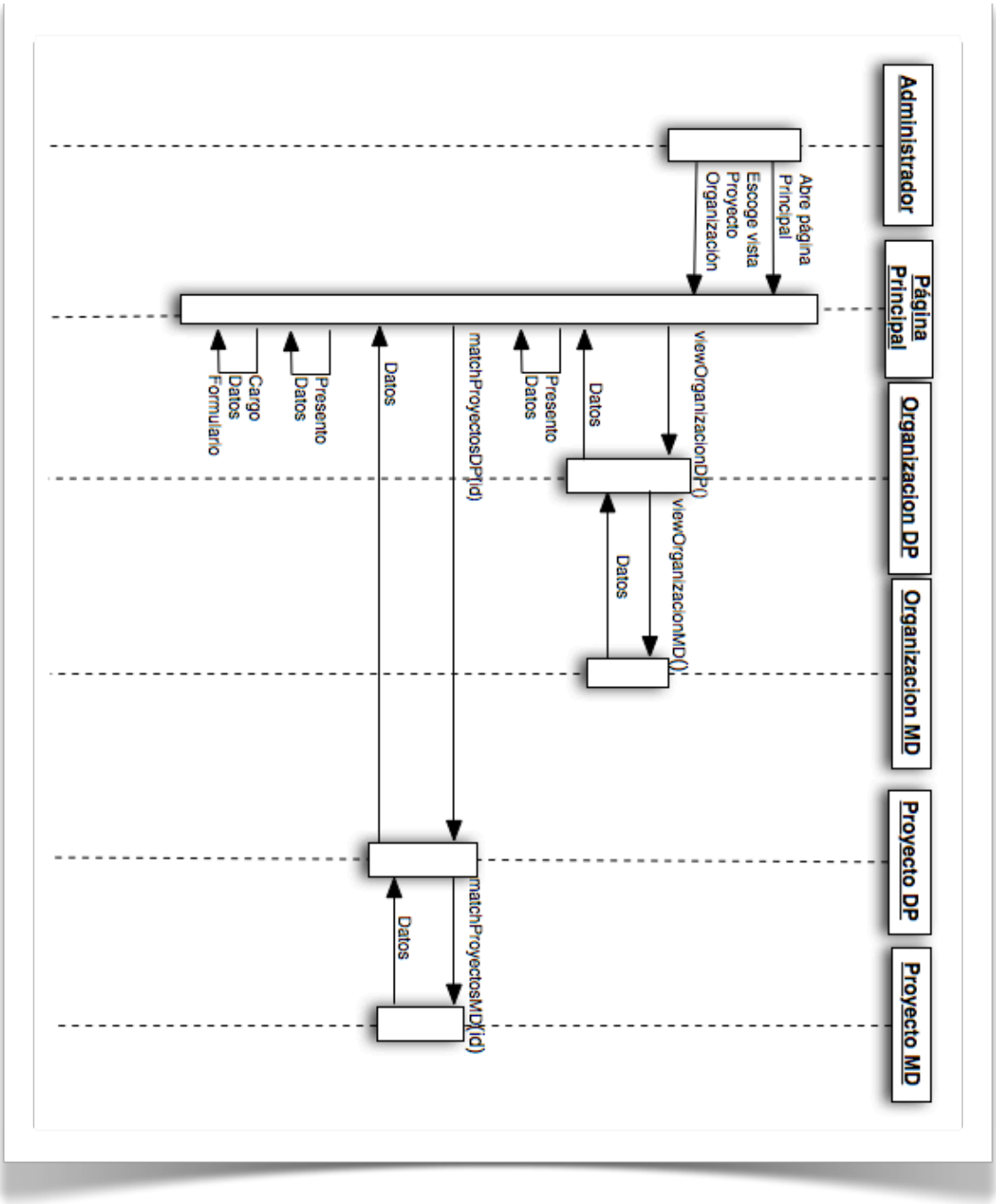
F 4.4 Eliminar Proyecto.



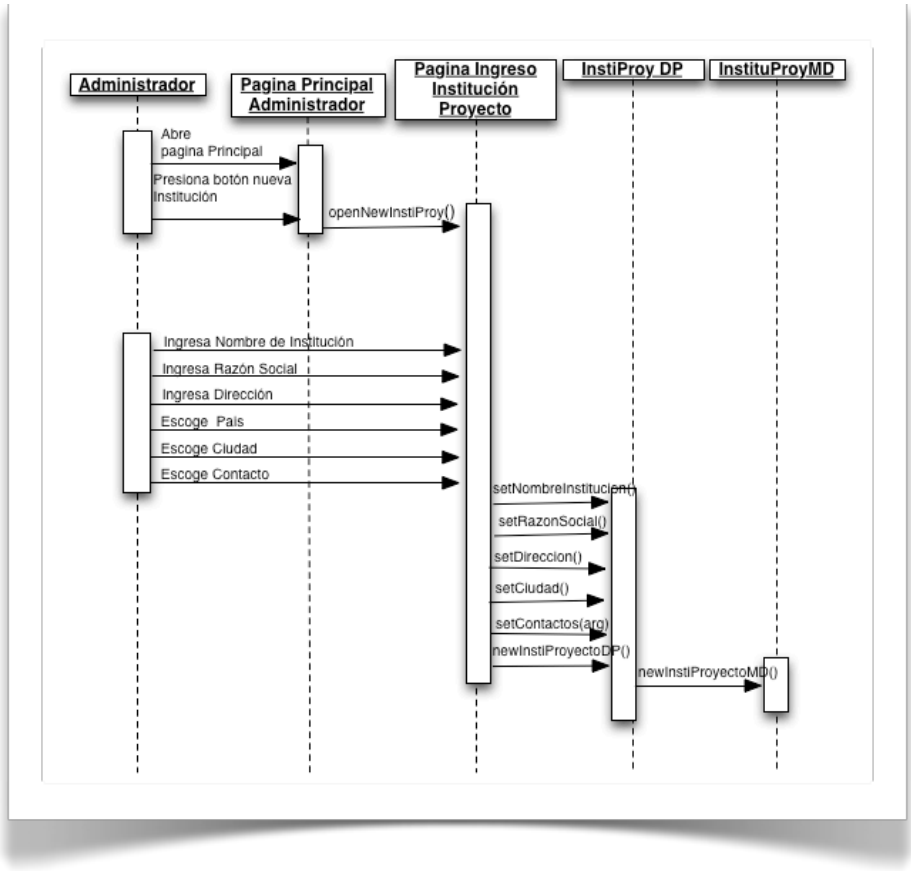
F 6 Asociar Proyecto Organización.



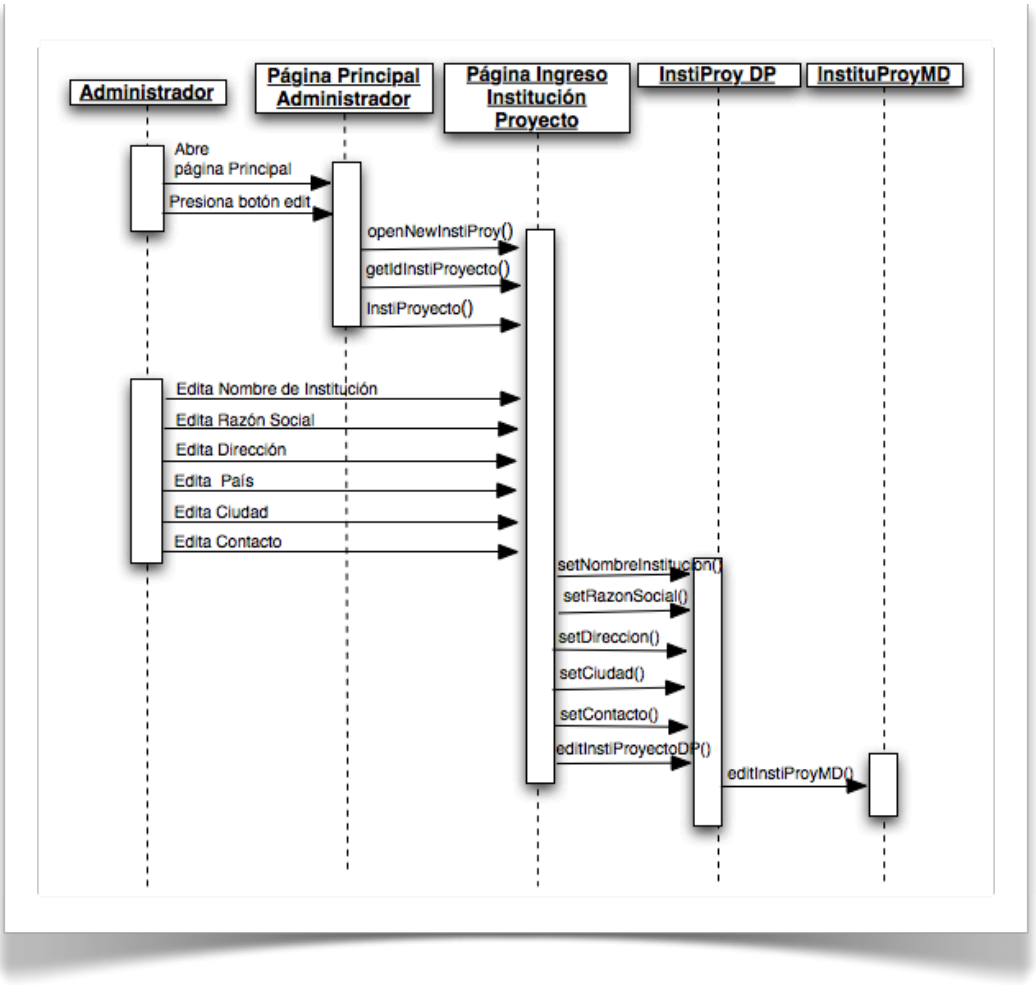
F 6 Asociar Organización Proyecto.



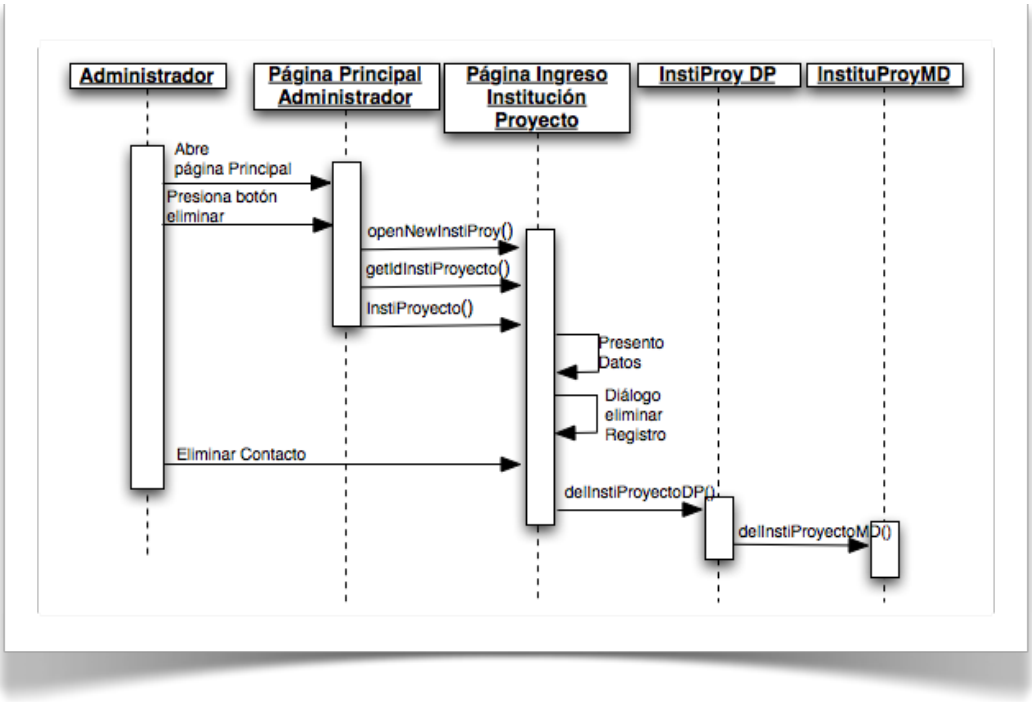
F 4.2 Nueva Institución Proyecto.



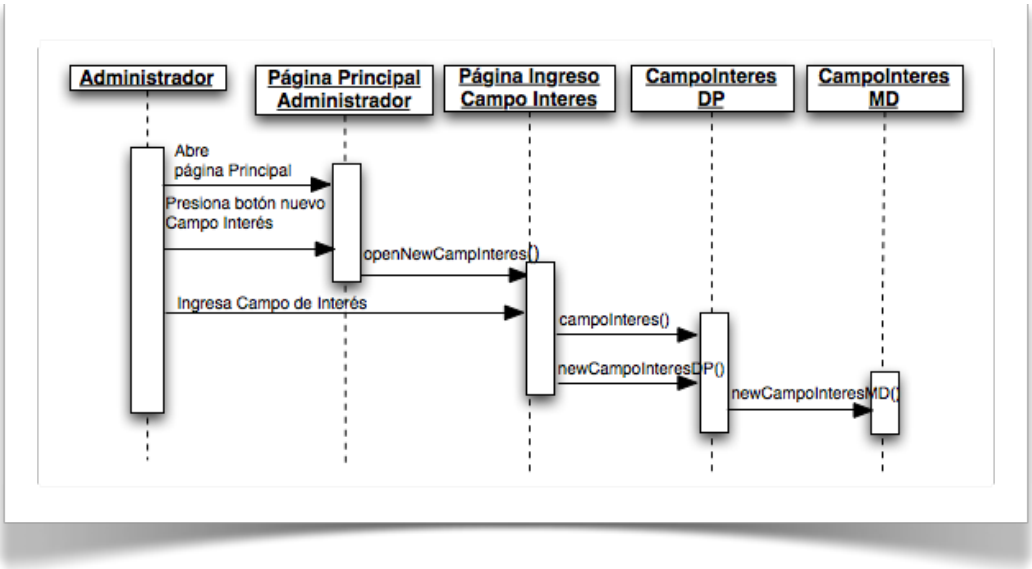
F 4.3 Editar Institución Proyecto.



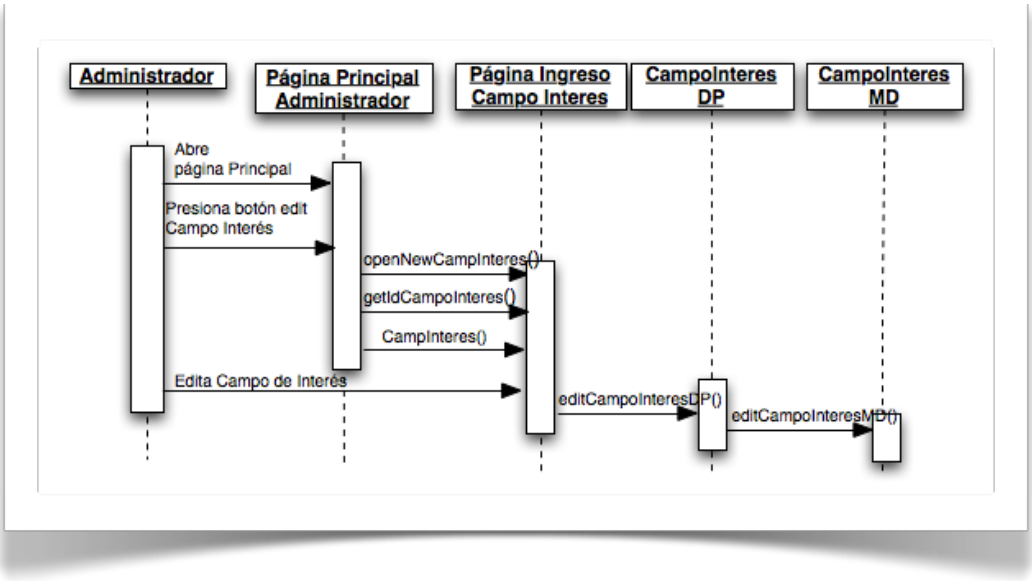
F 4.4 Eliminar Institución Proyecto.



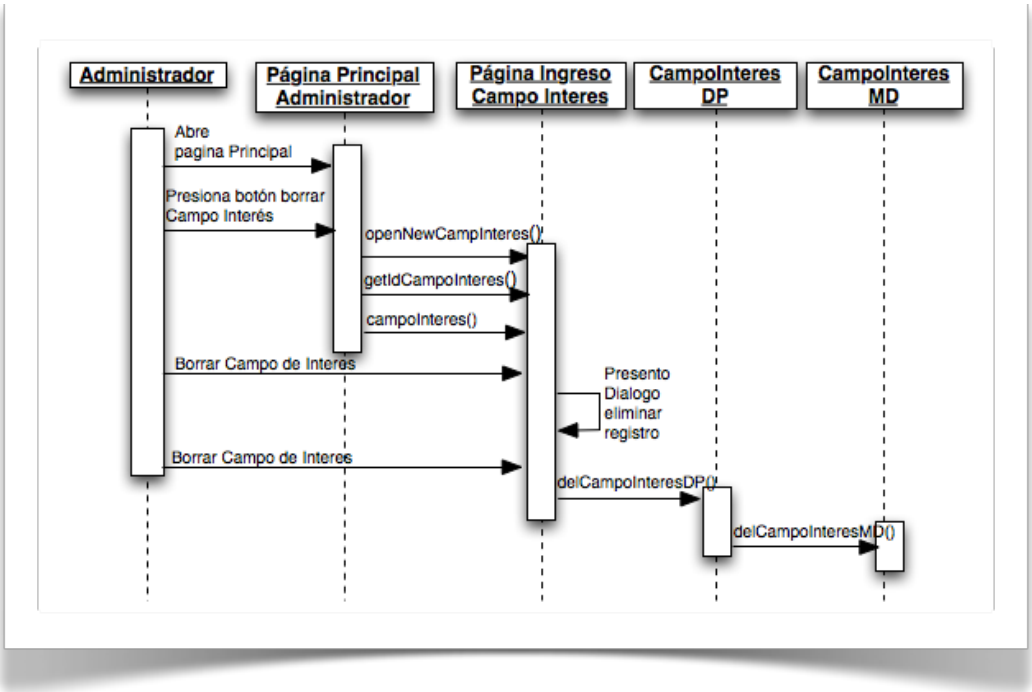
F 3.2 Nuevo Campo de Interés.



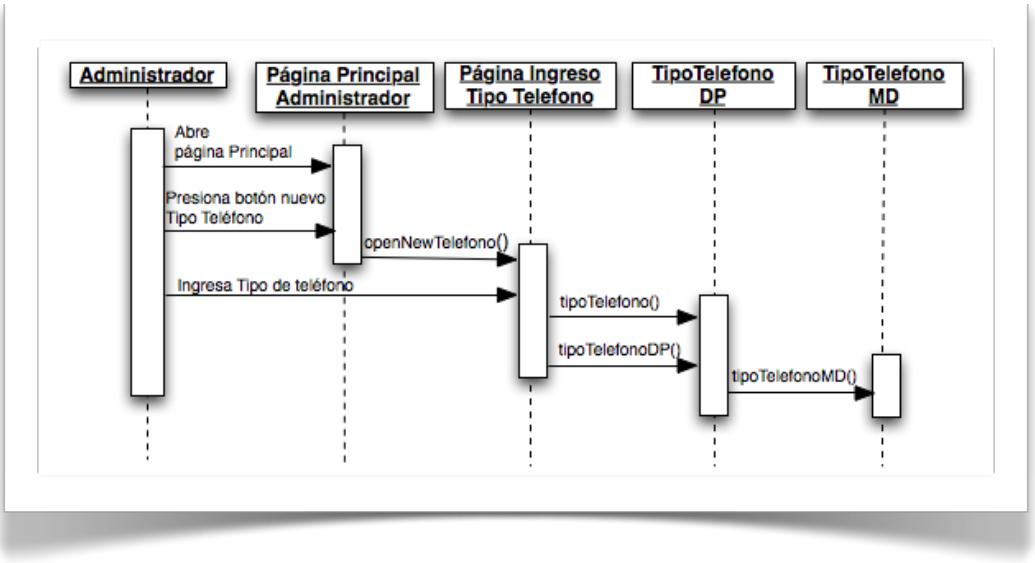
F 3.3 Editar Campo de Interés.



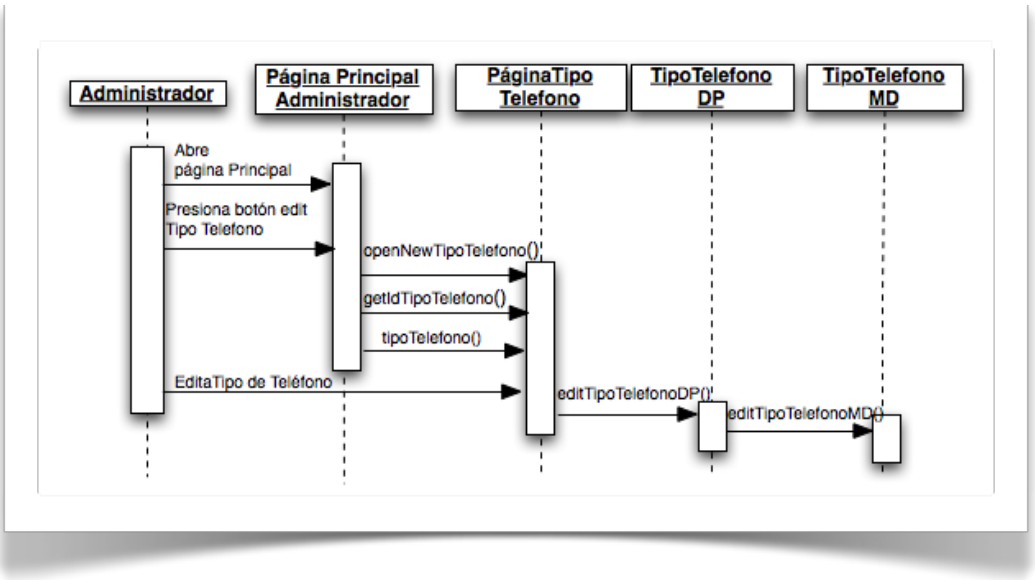
F 3.4 Eliminar Campo de Interés.



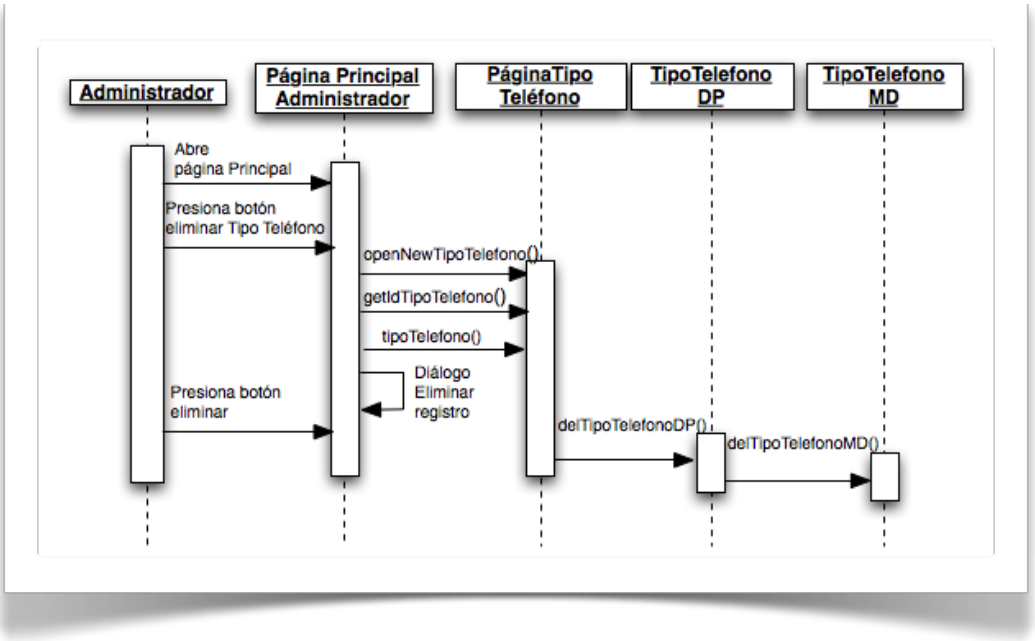
F 2.2 Nuevo Tipo de Teléfono.



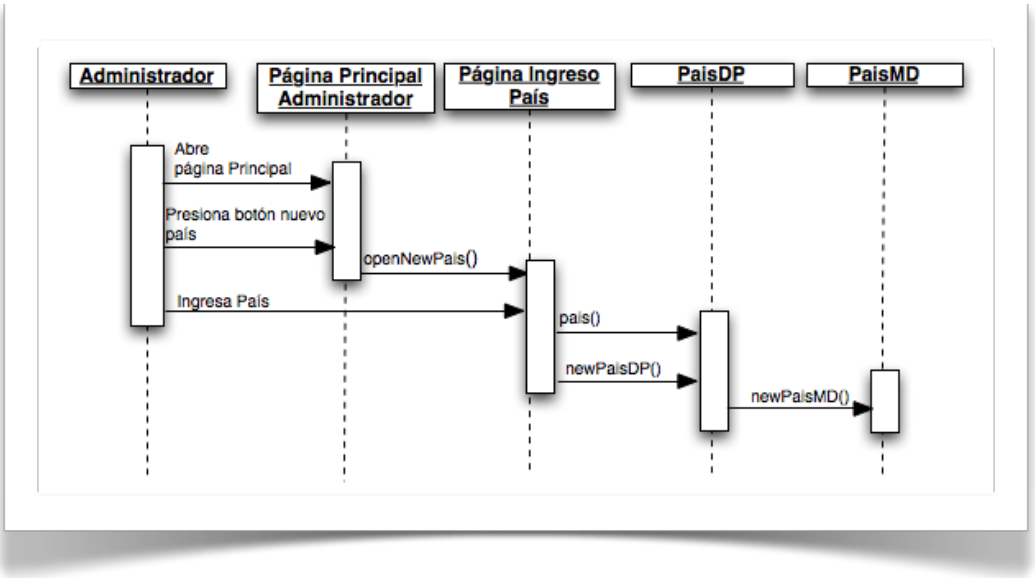
F 2.4 Editar Tipo de Teléfono.



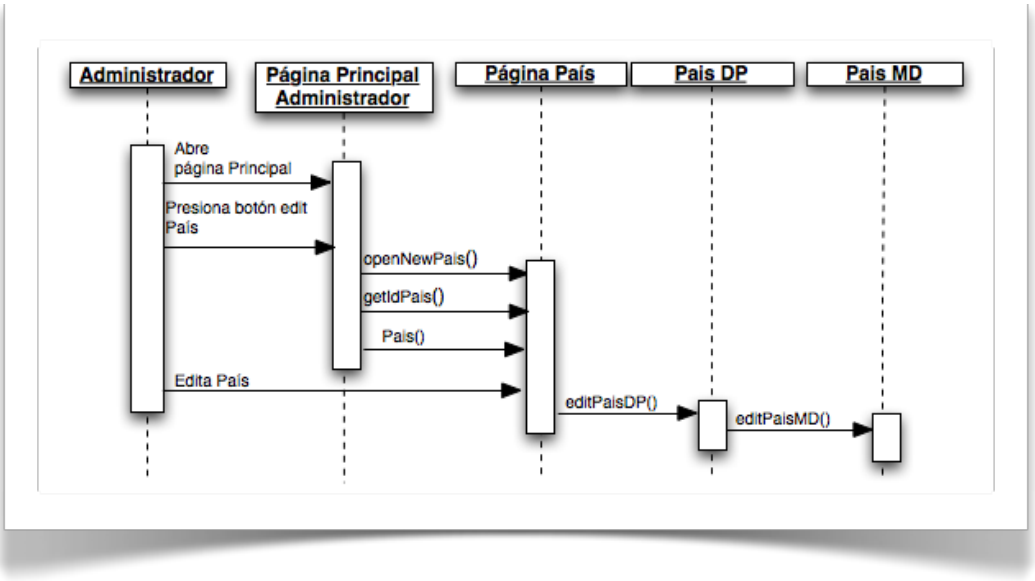
F2.3 Eliminar Tipo de Teléfono.



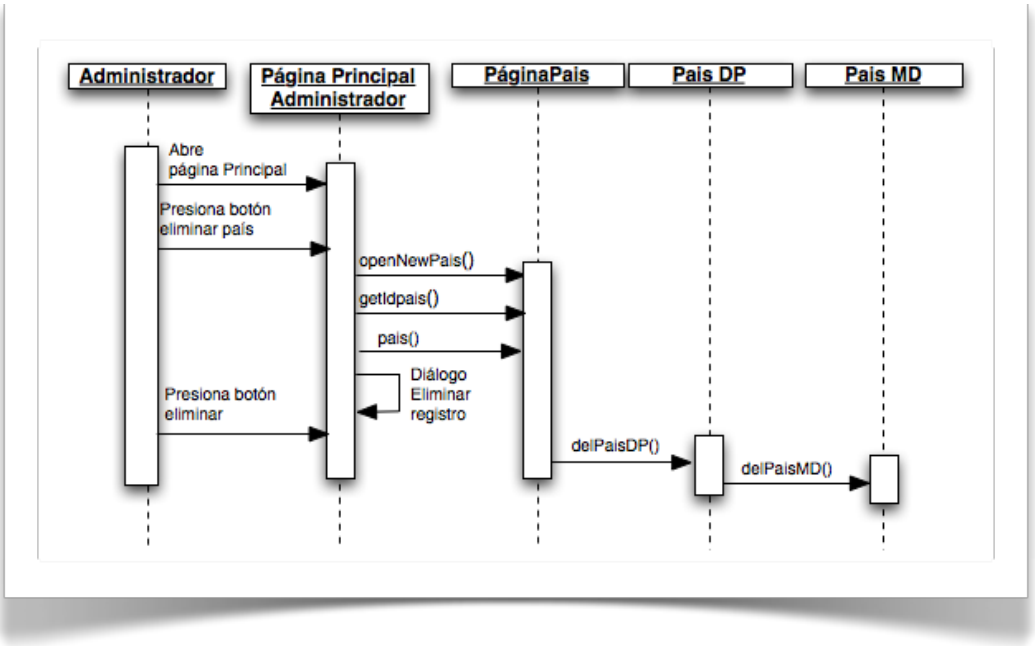
F 2.2 Nuevo País.



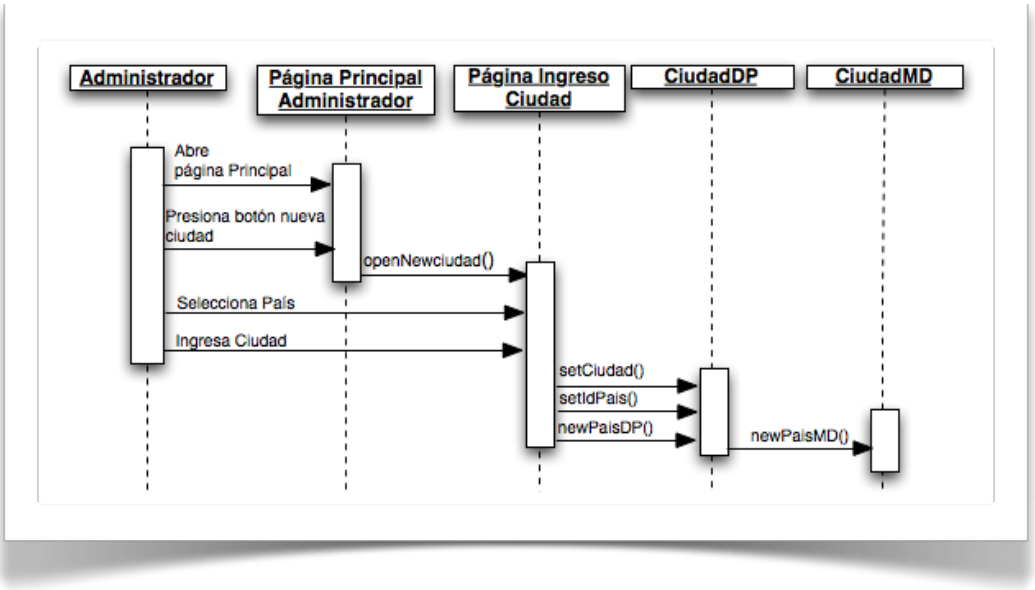
F 2.4 Editar País.



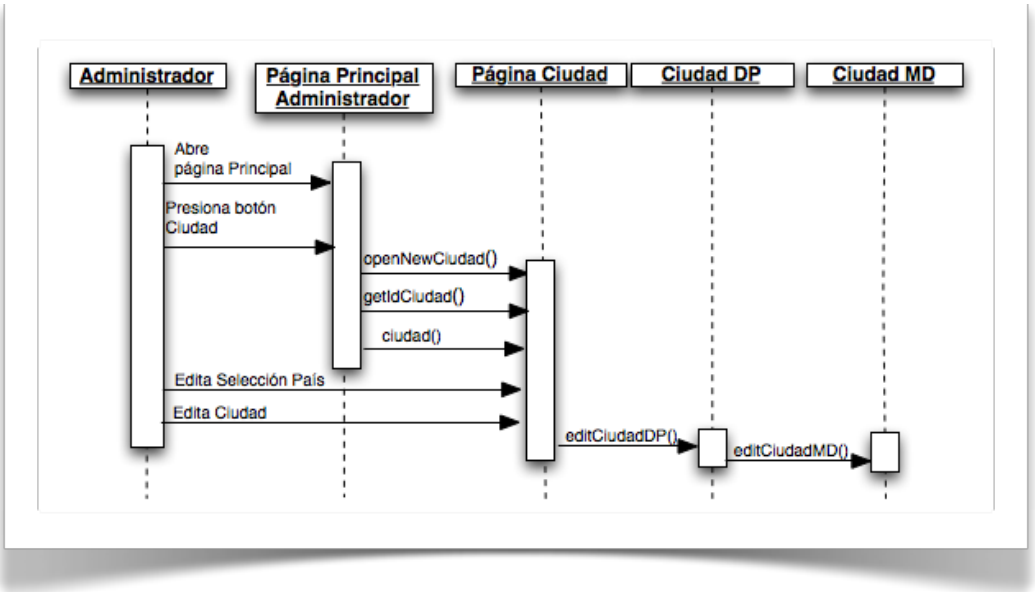
F 2.3 Eliminar País.



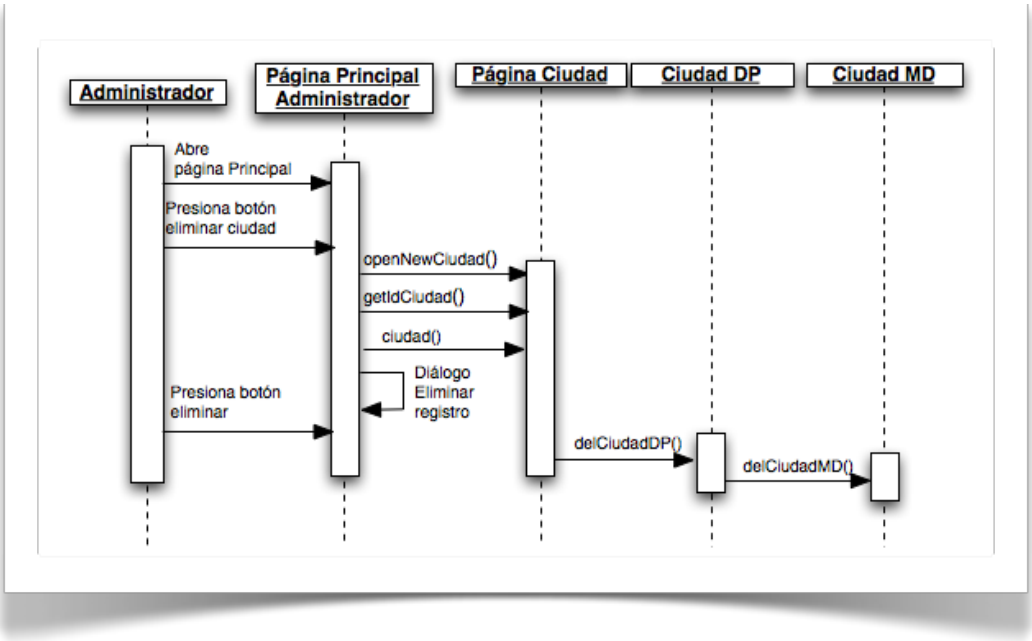
F 2.2 Nueva Ciudad.



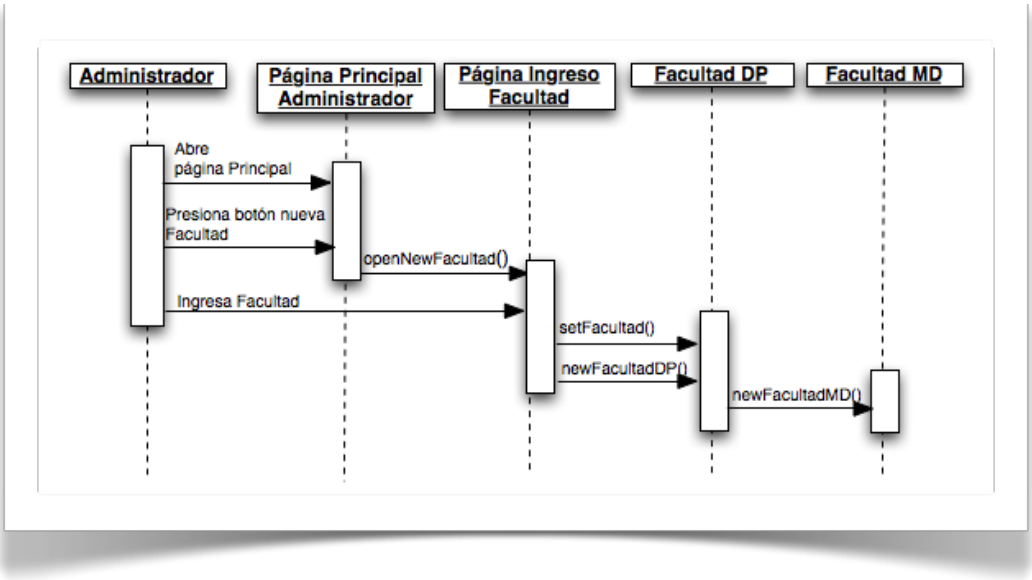
F 2.2 Editar Ciudad.



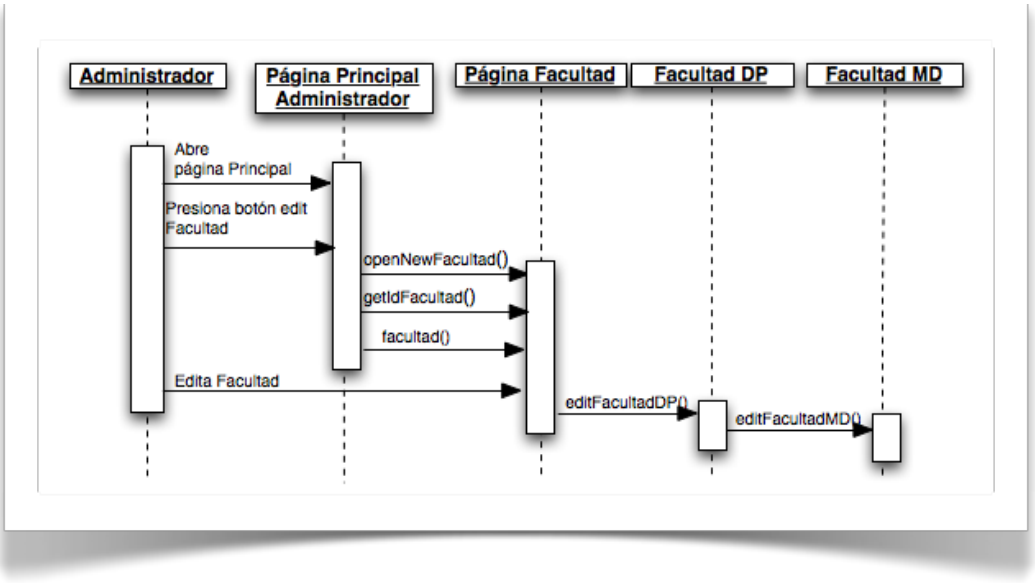
F 2.3 Eliminar Ciudad.



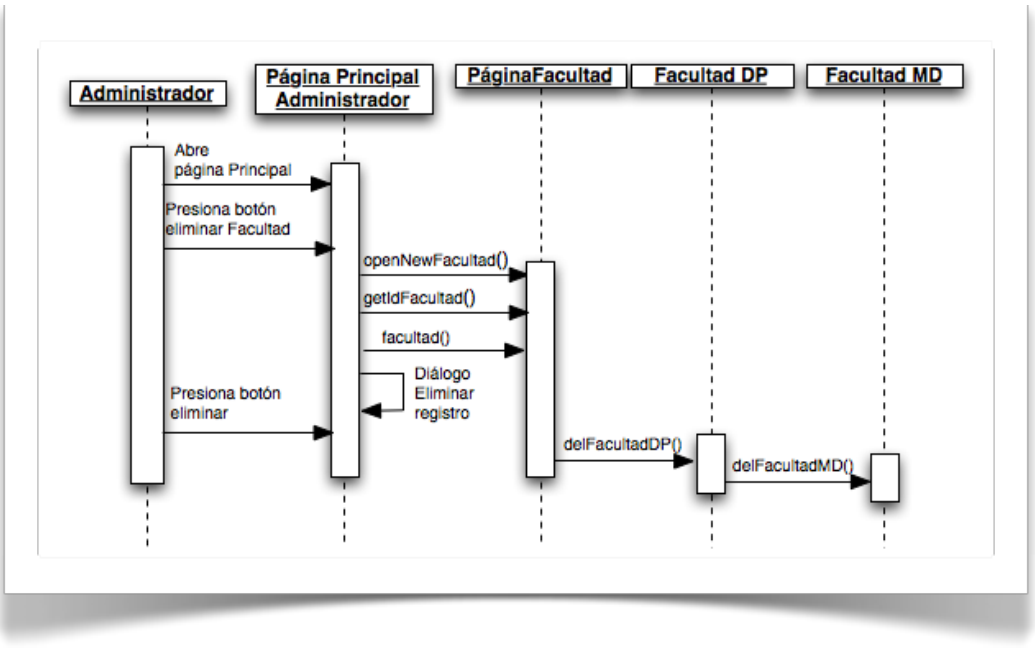
F 2.2 Nueva Facultad.



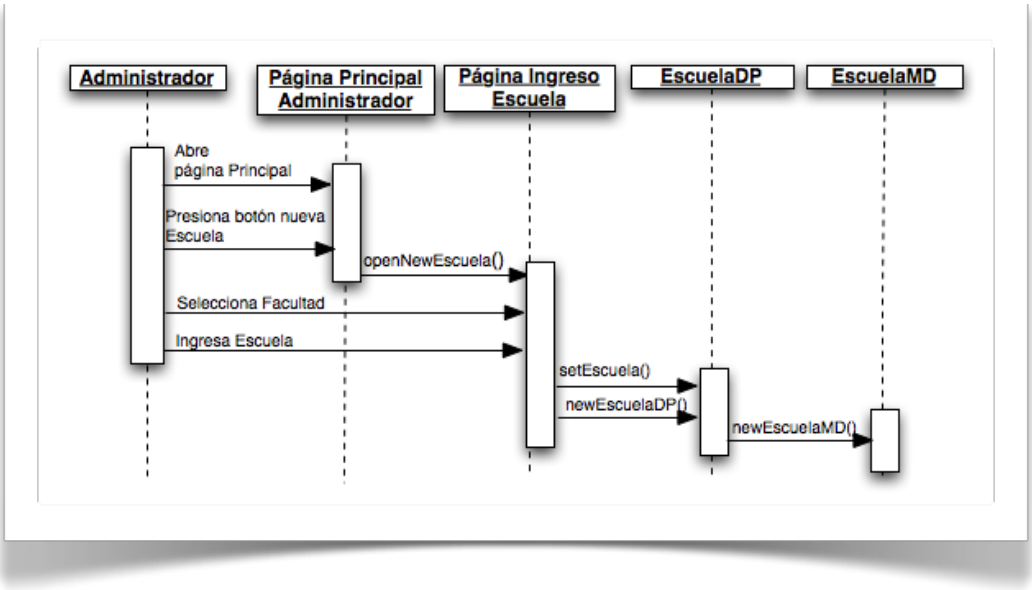
F 2.4 Editar Facultad.



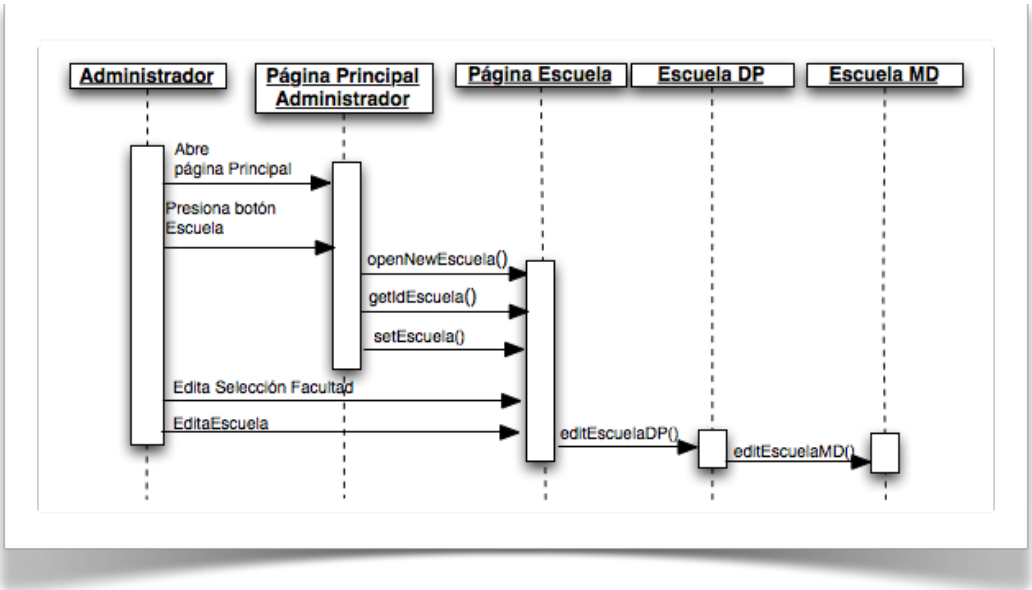
F 2.3 Eliminar Facultad.



F 2.3 Nueva Escuela.



F 2.4 Editar Escuela.



F 2.3 Eliminar Escuela.

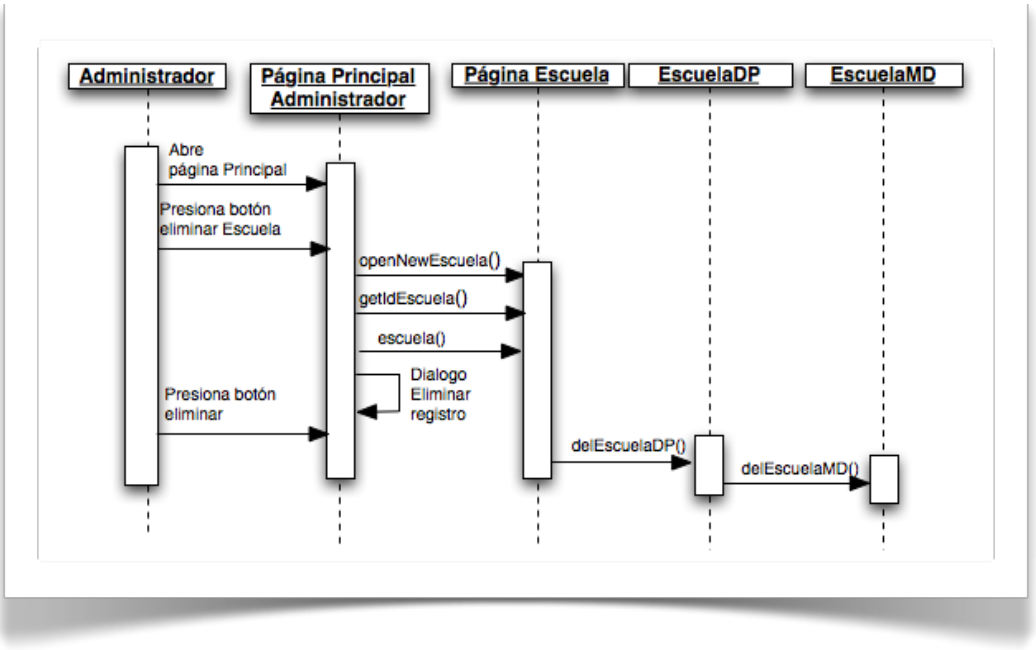
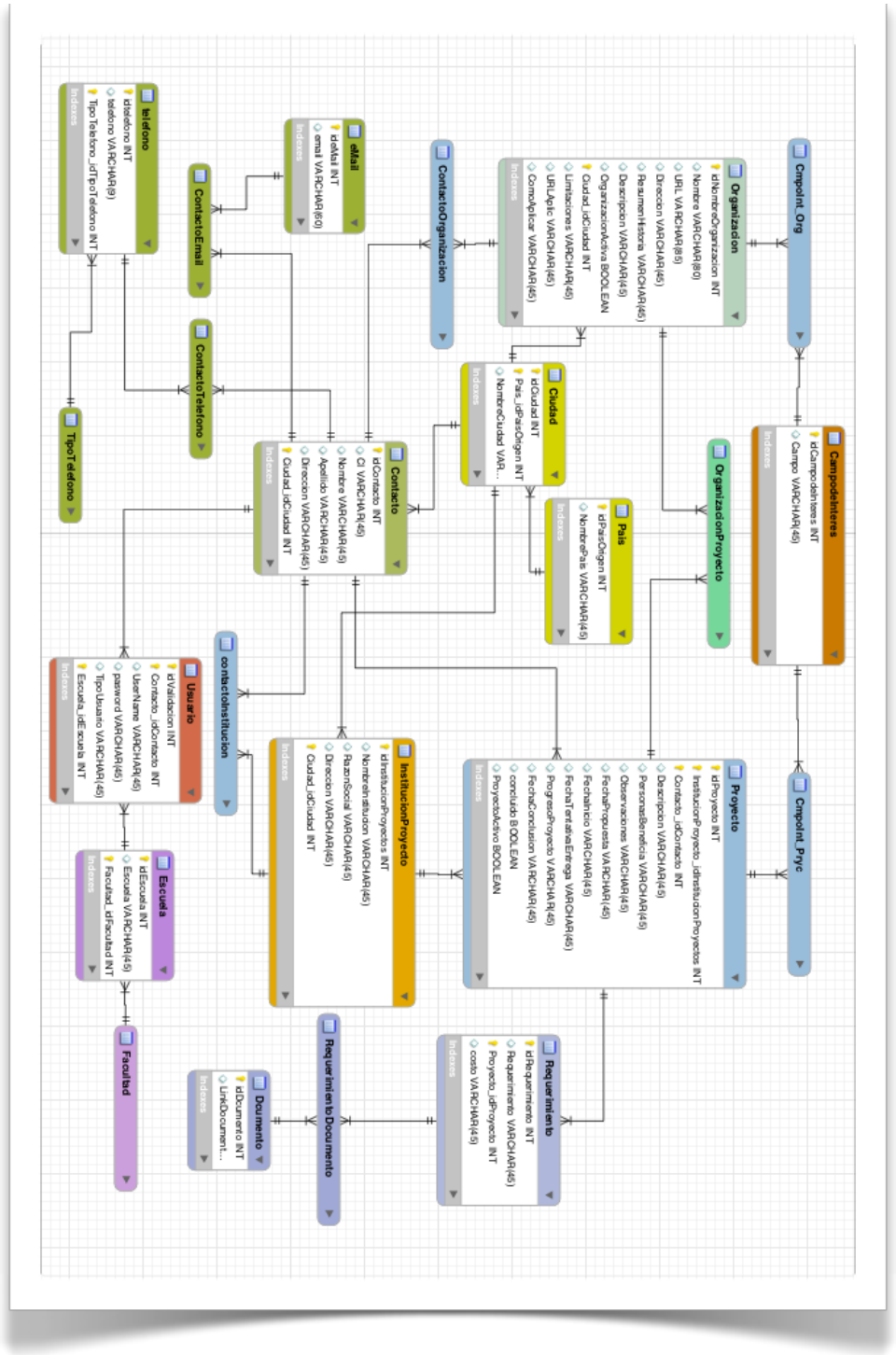


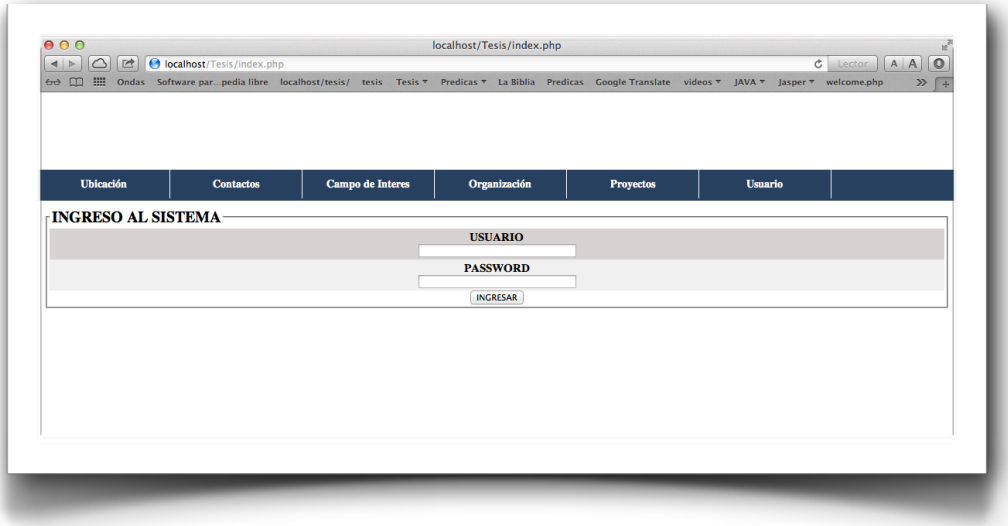
Diagrama de Base de Datos.

Se muestra a continuación, la representación gráfica de la base de datos, en la cual se observan las relaciones y los datos de las tablas.



Interfaz Gráfica

Inicio del Sistema

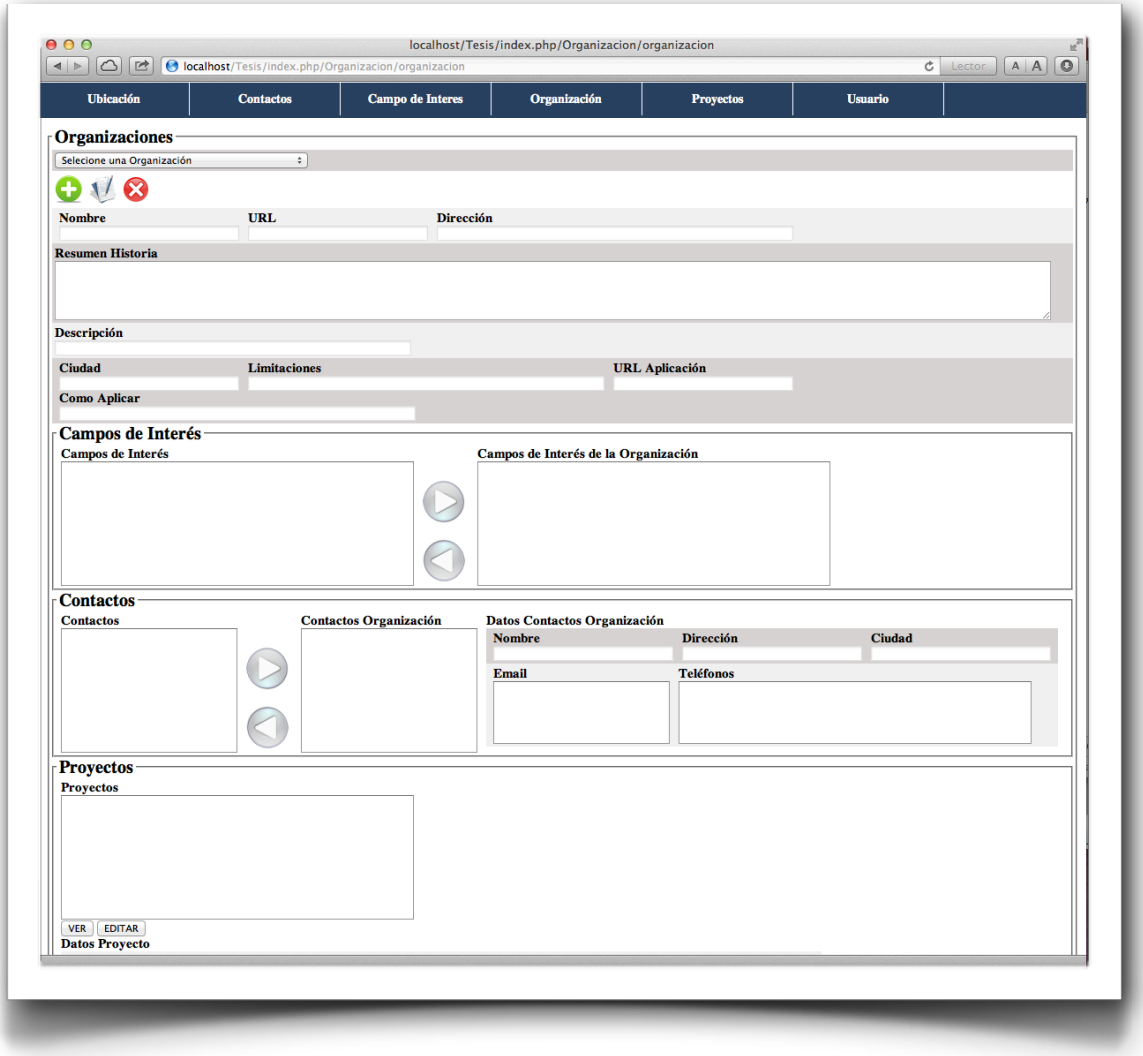


En la pantalla de inicio se muestran los campos de nombre de usuario y de contraseña, para permitir el ingreso a Usuarios y Administradores.

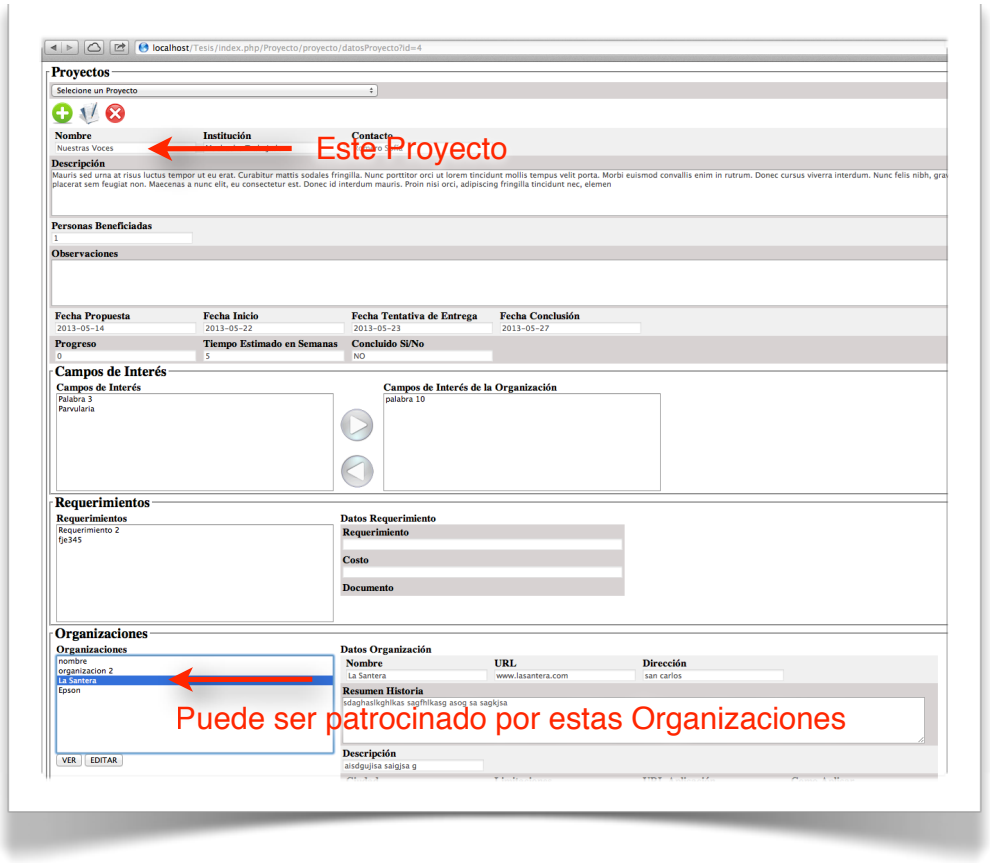
Página Principal

Una vez validado el tipo de usuario se muestra la página principal. Es la pantalla principal o por defecto, aquí es donde el Administrador o el Usuario van a utilizar la aplicación.

En administración el Usuario de la aplicación puede ingresar editar eliminar o actualizar los campos Teléfono, País, Ciudad entre Otros.



En Procesos el usuario podrá relacionar los proyectos con la organización y ademas



filtrar los proyectos y organizaciones mediante los campos de interés.

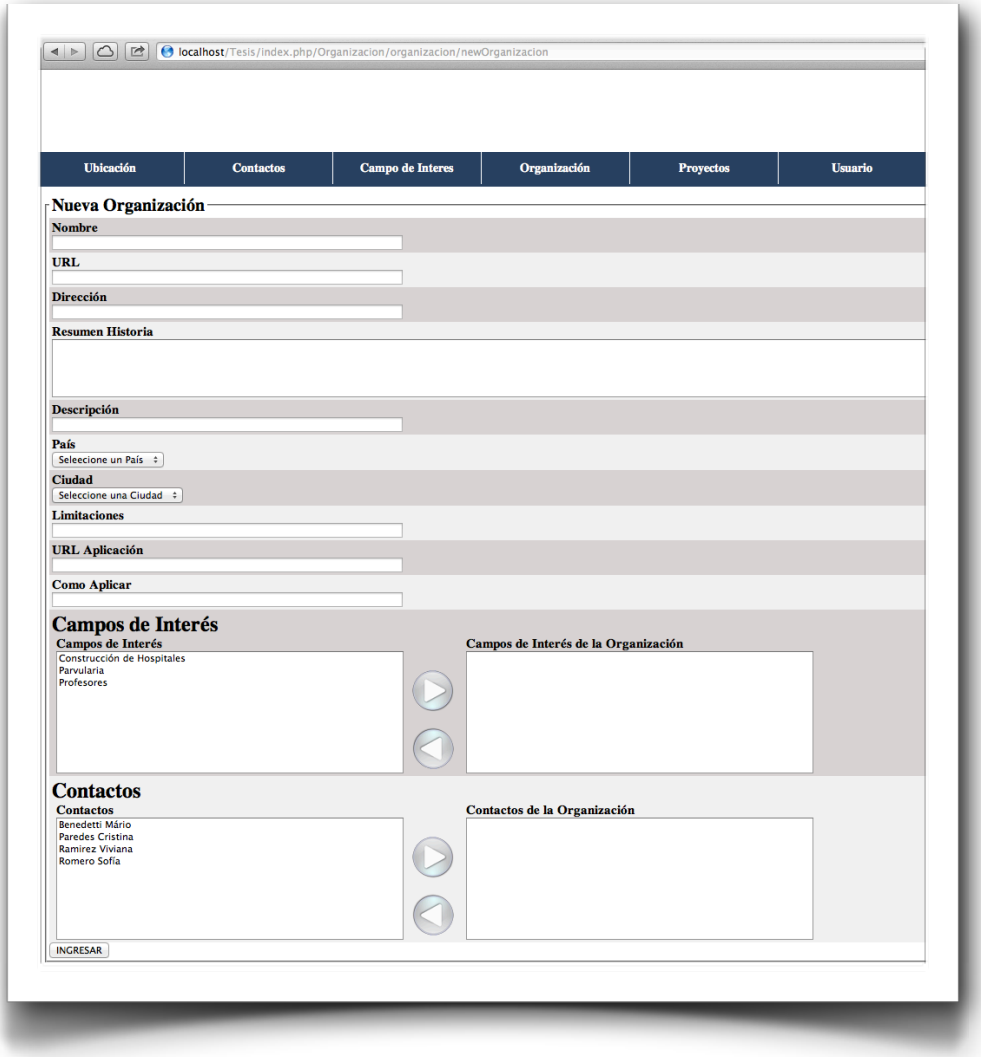
La administración de registros funciona de manera similar para los distintos registros

Nuevo

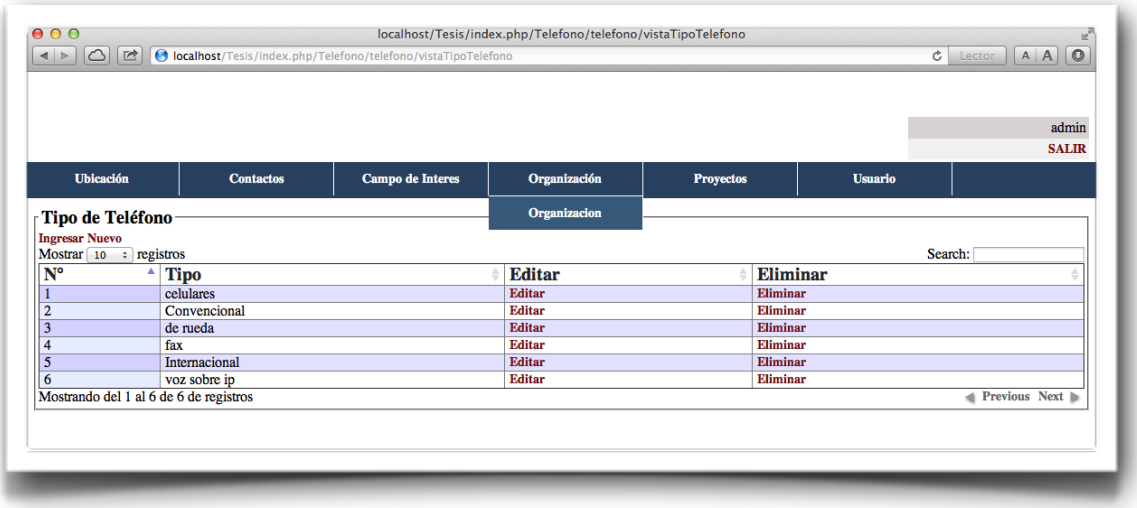
Para crear un nuevo Registro se debe presionamos el botón nuevo, entonces se presenta el formulario en blanco .

Editar

Para editar un registro, que se relaciona con mas de una tabla, como es el caso de "Organización" que carga información de la tabla ciudad, usuario entre otros se escoge el registro a editar del combo box con lo cual se carga los datos del registro se Presiona el botón editar y se carga el formulario con los datos actuales permitiendo actualizarlos.

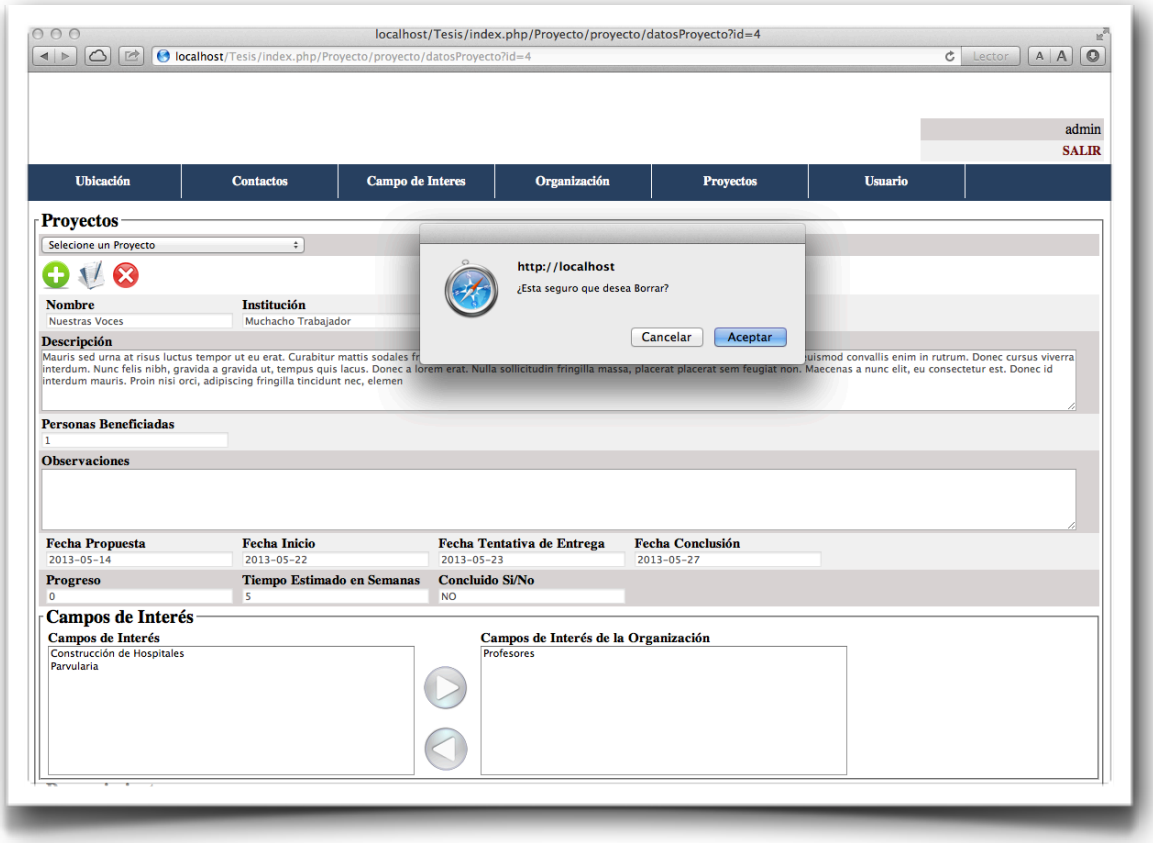


En el caso de registros que afectan a una solo tabla se muestra todos los datos a manera de lista en donde en la parte superior del listado se puede agregar registros y en la parte lateral izquierda del registro se puede editar o eliminar el registro.



Delete

Si el registro afecta a múltiples tablas seleccionar el registro deseado en el combo box para después cargar la información del registro en formulario. Presionar el botón borrar. Se pedirá confirmar la eliminación del registro antes de borrarlo.



Código Fuente (Actualizar Organización)
vista

El código que se muestra a continuación, en su mayoría es código html, pertenece a la capa de la vista, es la capa encargada de mostrar los datos al usuario de manera que lo puede entender, de igual manera recibe los datos y eventos por parte del usuario para ejecutar nuevos procesos.

updateOrganizacion.php[1]

```
<?php
$datos = array();
$datos = $dato[0];
?>

<fieldset>
    <legend>Editar Organización</legend>
    <form name="formulario" id="formulario" method="post" action="<?php
echo base_url().index_page();?>/Organizacion/organizacion/update">
        <div class="divItem1">
            <p>Nombre</p>
            <input class="imputGrande" type="text" name="nombre"
id="nombre" value="<?php echo $datos['nombre']; ?>">
        </div>
        <div class="divItem2">
            <p>URL</p>
            <input class="imputGrande" type="text" name="url" id="url"
value="<?php echo $datos['url']; ?>">
        </div>
        <div class="divItem1">
            <p>Dirección</p>
            <input  class="imputGrande"type="text" name="direccion"
id="direccion" value="<?php echo $datos['direccion']; ?>">
        </div>
        <div class="divItem2">
            <p>Resumen Historia</p>
            <textarea rows="5" name="resumen" id="resumen"><?php echo
$datos['resumen']; ?></textarea>
        </div>
        <div class="divItem1">
            <p>Descripción</p>
            <input class="imputGrande" type="text" name="descripcion"
id="descripcion" value="<?php echo $datos['descripcion']; ?>">
        </div>
        <div class="divItem2">
            <p>País</p>
            <select name="comboPais" id="comboPais"
onChange="verComboCiudad()">
                <option value="0">Seleecione un País</option>
                <?php foreach ($datosPais as $datosPaises) { ?>
```

[1] El código fuente de toda la aplicación esta incluido en el CD

```
                    <option value="<?php echo $datosPaises['id']; ?>" <?
php if ($idPais == $datosPaises['id']) { ?>selected="selected" <?php } ?
>><?php echo $datosPaises['nombre']; ?></option>
                <?php } ?>
            </select>
        </div>
        <div class="divItem1">
            <p>Ciudad</p>
            <select name="comboCiudad" id="comboCiudad">
                <option value="0">Seleccione una Ciudad</option>
                <?php foreach ($ciudad as $datosCiudad) { ?>
                    <option value="<?php echo $datosCiudad['id']; ?>" <?
php if ($datos['ciudad'] == $datosCiudad['id']) { ?>selected="selected"
<?php } ?> ><?php echo $datosCiudad['nombre']; ?></option>
                <?php } ?>
            </select>
        </div>
        <div class="divItem2">
            <p>Limitaciones</p>
            <input class="imputGrande" type="text" name="limitaciones"
id="limitaciones" value="<?php echo $datos['limitaciones']; ?>">
        </div>
        <div class="divItem1">
            <p>URL Aplicación</p>
            <input class="imputGrande" type="text" name="urlAplicacion"
id="urlAplicacion"  value="<?php echo $datos['urlAplicacion']; ?>">
        </div>
        <div class="divItem2">
            <p>Como Aplicar</p>
            <input  class="imputGrande" type="text" name="comoAplicar"
id="comoAplicar" value="<?php echo $datos['comoAplicar']; ?>">
        </div>
         <input type="hidden" id="idOrganizacion" name="idOrganizacion"
value="<?php echo $_GET['id']; ?>"/>
        <input type="submit" id="ingresar" value="INGRESAR" />

    </form>
</fieldset>
```

Controlador

El Controlador es el encargado de interactuar con el modelo y con la vista, sabe que hay que hacer pero no sabe como hacerlo, quien sabe como hacerlo es el modelo por lo tanto se encarga de canalizar la información entre el modelo y la vista de una manera preestablecida.

organizacion.php

```
<?php

class organizacion extends CI_Controller {

    public function index() {
        $this->load->helper('url');
```

```
        $this->load->model('classOrganizacion');
        $datos['organizacion'] = $this->classOrganizacion->verTodos();
        $this->load->view('header');
        $this->load->view('Organizacion/vistaOrganizacion', $datos);
        $this->load->view('footer');
    }

    public function datosOrganizacion() {
        $this->load->helper('url');
        $this->load->model('classOrganizacion');
        $this->load->model('classContacto');
        $this->load->model('classCampoInteres');
        $datosContacto = $this->classContacto-
>verContactosOrganizacion($_GET['id']);
        $datos['contactoNoOrganizacion'] = $this->classContacto-
>verContactosNoOrganizacion($_GET['id']);
        $datos['contactoOrganizacion'] = $datosContacto;
        $result = $this->classOrganizacion->verUnico($_GET['id']);
        $datos['organizacion'] = $this->classOrganizacion->verTodos();
        $datos['campoInteresNoOrganizacion'] = $this->classCampoInteres-
>verCamposNoOrganizacion($_GET['id']);
        $datos['campoInteresOrganizacion'] = $this->classCampoInteres-
>verCamposOrganizacion($_GET['id']);
        $datos['proyectoOrganizacion']=  $this->classCampoInteres-
>verProyectosOrganizacion($_GET['id']);
        if (mysql_num_rows($result) > 0) {
            $datosOrganizacion = array();
            while ($row = mysql_fetch_array($result)) {
                $datosOrganizacion['organizacion'] = $row[1];
                $datosOrganizacion['url'] = $row[2];
                $datosOrganizacion['direccion'] = $row[3];
                $datosOrganizacion['resumenHistoria'] = $row[4];
                $datosOrganizacion['descripcion'] = $row[5];
                $datosOrganizacion['organizacionActiva'] = $row[6];
                $datosOrganizacion['ciudad'] = $row[11];
                $datosOrganizacion['limitaciones'] = $row[8];
                $datosOrganizacion['URLAplicacion'] = $row[9];
                $datosOrganizacion['comoAplicar'] = $row[10];
            }
            $datos['datosOrganizacion'] = $datosOrganizacion;
            $this->load->view('header');
            $this->load->view('Organizacion/vistaOrganizacion', $datos);
            $this->load->view('footer');
        } else {
            $this->load->view('header');
            $this->load->view('error');
            $this->load->view('footer');
        }
    }

    public function newOrganizacion() {
        $this->load->helper('url');
        $this->load->model('classPais');
        $this->load->model('classCiudad');
        $this->load->model('classCampoInteres');
        $this->load->model('classContacto');
        $datosPais = $this->classPais->verTodos();
```

```
        $datos['datosPais'] = $datosPais;
        $datos['campoInteres'] = $this->classCampoInteres->verTodos();
        $datos['contacto'] = $this->classContacto->verTodos();
        $this->load->view('header');
        $this->load->view('Organizacion/newOrganizacion', $datos);
        $this->load->view('footer');
    }

    public function updateOrganizacion() {
        $idCiudad = 0;
        $idPais = 0;
        $this->load->helper('url');
        $this->load->model('classOrganizacion');
        $this->load->model('classCiudad');
        $this->load->model('classPais');
        $sql = "select * from organizacion where idOrganizacion=" .
$_GET['id'];
        $resultOrganizacion = $this->classOrganizacion->select($sql);
        $datos = array();
        while ($row = mysql_fetch_array($resultOrganizacion)) {
            $datos[] = array(
                'idOrganizacion' => $row[0],
                'nombre' => $row[1],
                'url' => $row[2],
                'direccion' => $row[3],
                'resumen' => $row[4],
                'descripcion' => $row[5],
                'organizacionActiva' => $row[6],
                'ciudad' => $row[7],
                'limitaciones' => $row[8],
                'urlAplicacion' => $row[9],
                'comoAplicar' => $row[10],
            );
            $idCiudad = $row[7];
        }
        $sql = "select Pais_idPais from ciudad where idCiudad=" .
$idCiudad;
        $resultCiudad = $this->classCiudad->select($sql);
        while ($row = mysql_fetch_array($resultCiudad)) {
            $idPais = $row[0];
        }
        $sql = "select * from ciudad where Pais_idPais=" . $idPais . "
order by NombreCiudad";
        $resultCiudad = $this->classCiudad->select($sql);
        $datosCiudad = array();
        while ($row = mysql_fetch_array($resultCiudad)) {
            $datosCiudad[] = array(
                'id' => $row[0],
                'nombre' => $row[2]
            );
        }
        $datosPais = $this->classPais->verTodos();
        $datosFinales['idPais'] = $idPais;
        $datosFinales['datosPais'] = $datosPais;
        $datosFinales['dato'] = $datos;
        $datosFinales['ciudad'] = $datosCiudad;
        $this->load->view('header');
```

```
        $this->load->view('Organizacion/updateOrganizacion',
$datosFinales);
        $this->load->view('footer');
    }

    public function insert() {
        $this->load->helper('url');
        $this->load->model('classOrganizacion');
        $this->classOrganizacion->setNombre($_POST['nombre']);
        $this->classOrganizacion->setURL($_POST['url']);
        $this->classOrganizacion->setDireccion($_POST['direccion']);
        $this->classOrganizacion->setResumenHistoria($_POST['resumen']);
        $this->classOrganizacion->setDescripcion($_POST['descripcion']);
        $this->classOrganizacion->setOrganizacionActiva(1);
        $this->classOrganizacion-
>setLimitaciones($_POST['limitaciones']);
        $this->classOrganizacion->setURLAplic($_POST['urlAplicacion']);
        $this->classOrganizacion->setComoAplicar($_POST['comoAplicar']);
        $this->classOrganizacion->insert($_POST['comboCiudad']);
        $datos = $this->classOrganizacion->verUltimo();
        foreach ($datos as $dato) {
            $idOrganizacion = $dato['id'];
        }
        $arregloContacto = explode(",", $_POST['arregloContactos']);
        $arregloCampo = explode(",", $_POST['arregloCamposInteres']);
        for ($i = 0; $i < count($arregloContacto); $i++) {
            $datos = array(
                'Organizacion_idOrganizacion' => $idOrganizacion,
                'Contacto_idContacto' => $arregloContacto[$i]
            );
            $this->classOrganizacion->insertContactoOrganizacion($datos);
        }
        for ($i = 0; $i < count($arregloCampo); $i++) {
            $datos = array(
                'Organizacion_idOrganizacion' => $idOrganizacion,
                'CampoInteres_idCampoInteres' => $arregloCampo[$i]
            );
            $this->classOrganizacion->insertCamposOrganizacion($datos);
        }
        header('location:' . base_url() . index_page() . '/Organizacion/
organizacion/datosOrganizacion?id=' . $idOrganizacion);
    }

    public function update() {
        $where = array(
            'idOrganizacion' => $_POST['idOrganizacion']
        );
        $this->load->helper('url');
        $this->load->model('classOrganizacion');
        $this->classOrganizacion->setNombre($_POST['nombre']);
        $this->classOrganizacion->setURL($_POST['url']);
        $this->classOrganizacion->setDireccion($_POST['direccion']);
        $this->classOrganizacion->setResumenHistoria($_POST['resumen']);
        $this->classOrganizacion->setDescripcion($_POST['descripcion']);
        $this->classOrganizacion->setOrganizacionActiva(1);
        $this->classOrganizacion-
>setLimitaciones($_POST['limitaciones']);
```

```
        $this->classOrganizacion->setURLAplic($_POST['urlAplicacion']);
        $this->classOrganizacion->setComoAplicar($_POST['comoAplicar']);

        $this->classOrganizacion->update($_POST['comboCiudad'], $where);
        header('location:'.base_url().index_page().'/Organizacion/
organizacion/datosOrganizacion?id='.$_POST['idOrganizacion']);
    }

    public function delete(){
        $where = array(
            'idOrganizacion' => $_POST['idOrganizacion']
        );
        $this->load->helper('url');
        $this->load->model('classOrganizacion');
        $this->classOrganizacion->delete($where);
        echo json_encode($where);
    }

    public function insertCampoOrganizacion() {
        $this->load->helper('url');
        $this->load->model('classOrganizacion');
        $datos = array(
            'Organizacion_idOrganizacion' => $_POST['idOrganizacion'],
            'CampoInteres_idCampoInteres' => $_POST['idCampo']
        );
        $this->classOrganizacion->insertCamposOrganizacion($datos);
        echo json_encode($datos);
    }
    public function deleteCampoOrganizacion() {
        $this->load->helper('url');
        $this->load->model('classOrganizacion');
        $datos = array(
            'Organizacion_idOrganizacion' => $_POST['idOrganizacion'],
            'CampoInteres_idCampoInteres' => $_POST['idCampo']
        );
        $this->classOrganizacion->deleteCamposOrganizacion($datos);
        echo json_encode($datos);
    }

     public function verDatosOrganizacionProyecto(){
        $this->load->helper('url');
        $this->load->model('classOrganizacion');
        $result=  $this->classOrganizacion-
>verUnico($_POST['idOrganizacion']);
        while($row=  mysql_fetch_array($result)){
            $datos=array(
                'id'=>$row[0],
                'nombre'=>$row[1],
                'url'=>$row[2],
                'direccion'=>$row[3],
                'resumenHistoria'=>$row[4],
                'descripcion'=>$row[5],
                'limitaciones'=>$row[8],
                'urlAplicacion'=>$row[9],
                'comoAplicar'=>$row[10],
                'ciudad'=>$row[11]
            );
```

```
        }
        echo json_encode($datos);
    }


}

?>
```

Modelo

El Modelo dispone de todas las clases y métodos necesarios para satisfacer los requerimientos establecidos por el usuario, es el que maneja la lógica del negocio se conecta con bases de datos ínter actúa con otras clases a fin de responder a las solicitudes requeridas.

classOrganizacion.php

```
<?php

class classOrganizacion extends CI_Model {

    //put your code here
    private $Nombre;
    private $URL;
    private $Direccion;
    private $ResumenHistoria;
    private $Descripcion;
    private $OrganizacionActiva;
    private $Limitaciones;
    private $URLAplic;
    private $ComoAplicar;

    public function __construct() {
        parent::__construct();
        $this->Nombre = "";
        $this->URL = "";
        $this->Direccion = "";
        $this->ResumenHistoria = "";
        $this->Descripcion = "";
        $this->OrganizacionActiva = "";
        $this->Limitaciones = "";
        $this->URLAplic = "";
        $this->ComoAplicar = "";
    }

    public function getNombre() {

        return $this->Nombre;
    }
```

```
public function setNombre($Nombre) {

    $this->Nombre = $Nombre;
}

public function getURL() {

    return $this->URL;
}

public function setURL($URL) {

    $this->URL = $URL;
}

public function getDireccion() {

    return $this->Direccion;
}

public function setDireccion($Direccion) {

    $this->Direccion = $Direccion;
}

public function getResumenHistoria() {

    return $this->ResumenHistoria;
}

public function setResumenHistoria($ResumenHistoria) {

    $this->ResumenHistoria = $ResumenHistoria;
}

public function getDescripcion() {

    return $this->Descripcion;
}

public function setDescripcion($Descripcion) {

    $this->Descripcion = $Descripcion;
}

public function getOrganizacionActiva() {

    return $this->OrganizacionActiva;
}

public function setOrganizacionActiva($OrganizacionActiva) {

    $this->OrganizacionActiva = $OrganizacionActiva;
}

public function getLimitaciones() {
```

```
        return $this->Limitaciones;
    }

    public function setLimitaciones($Limitaciones) {

        $this->Limitaciones = $Limitaciones;
    }

    public function getURLAplic() {

        return $this->URLAplic;
    }

    public function setURLAplic($URLAplic) {

        $this->URLAplic = $URLAplic;
    }

    public function getComoAplicar() {

        return $this->ComoAplicar;
    }

    public function setComoAplicar($ComoAplicar) {

        $this->ComoAplicar = $ComoAplicar;
    }

    public function insert($idCiudad) {
        $this->load->model('classDB');
        $datos = array(
            'nombre' => $this->Nombre,
            'url' => $this->URL,
            'direccion' => $this->Direccion,
            'resumenHistoria' => $this->ResumenHistoria,
            'descripcion' => $this->Descripcion,
            'organizacionActiva' => $this->OrganizacionActiva,
            'Ciudad_idCiudad' => (int) $idCiudad,
            'limitaciones' => $this->Limitaciones,
            'URLAplicacion' => $this->URLAplic,
            'comoAplicar' => $this->ComoAplicar,
        );
        if ($this->classDB->insert('organizacion', $datos)) {
            return true;
        } else {
            return false;
        }
    }

    public function update($idCiudad, $where) {
        $this->load->model('classDB');
        $datos = array(
            'nombre' => $this->Nombre,
            'url' => $this->URL,
            'direccion' => $this->Direccion,
            'resumenHistoria' => $this->ResumenHistoria,
            'descripcion' => $this->Descripcion,
```

```
            'organizacionActiva' => $this->OrganizacionActiva,
            'ciudad_idCiudad' => $idCiudad,
            'limitaciones' => $this->Limitaciones,
            'URLAplicacion' => $this->URLAplic,
            'comoAplicar' => $this->ComoAplicar,
        );
        if ($this->classDB->update('organizacion', $datos, $where)) {
            return true;
        } else {
            return false;
        }
    }

    public function delete($where) {
        $this->load->model('classDB');
        $datos = array(
            'OrganizacionActiva' => 0
        );
        $this->classDB->update('organizacion', $datos, $where);
    }

    public function select($sql) {
        $this->load->model('classDB');
        $result = "";
        $result = $this->classDB->select($sql);
        return $result;
    }

    public function verTodos() {
        $sql = "select * from organizacion where OrganizacionActiva=1
order by nombre";
        $this->load->model('classDB');
        $result = "";
        $result = $this->classDB->select($sql);
        $datos = array();
        while ($row = mysql_fetch_array($result)) {
            $datos[] = array(
                'id' => $row[0],
                'organizacion' => $row[1]
            );
        }
        return $datos;
    }

    public function verUnico($id) {
        $sql = "select *,
            (SELECT nombreCiudad FROM ciudad WHERE
idCiudad=Ciudad_idCiudad) AS 'Ciudad'
            from organizacion
            where idOrganizacion=" . $id . " order by nombre";
        $this->load->model('classDB');
        $result = "";
        $result = $this->classDB->select($sql);
        return $result;
    }

    public function verUltimo() {
```

```
        $sql = "select * from organizacion order by idOrganizacion desc
limit 0,1";
        $this->load->model('classDB');
        $result = "";
        $result = $this->classDB->select($sql);
        $datos = array();
        while ($row = mysql_fetch_array($result)) {
            $datos[] = array(
                'id' => $row[0]
            );
        }
        return $datos;
    }

    public function insertContactoOrganizacion($datos) {
        $this->load->model('classDB');
        $this->classDB->insert('contactoorganizacion', $datos);
    }

    public function insertCamposOrganizacion($datos) {
        $this->load->model('classDB');
        $this->classDB->insert('cmpoint_org', $datos);
    }

    public function deleteCamposOrganizacion($where) {
        $this->load->model('classDB');
        $this->classDB->delete('cmpoint_org', $where);
    }

}

?>
```

CAPÍTULO 5

Se detallan las conclusiones y recomendaciones obtenidas en transcurso del desarrollo de la presente disertación de grado.

Conclusiones

El presente sistema, pretende encontrar el financiamiento para la mayor cantidad posible de proyectos disponibles, ya que se cuenta con un algoritmo capaz de filtrar la información común entre lo que las organizaciones pueden patrocinar y lo que los proyecto requieren de este modo se tiene una búsqueda más precisa, y se lleva un seguimiento de los proyectos y las organizaciones disponibles haciendo del sistema una herramienta idónea para el área de Acción Social.

La acertada decisión de obtener un directorio de Organizaciones en una base de datos nos permite manejar de manera sencilla grandes volúmenes de información de manera fácil enfocándonos en localizar la organización adecuada para cada proyecto.

El realizar la aplicación bajo un entorno web nos da la capacidad de usarla en cualquier parte con una conexión a internet y sobre casi cualquier sistema operativo que ejecute un navegador web que soporte la tecnología usada y hasta en dispositivos móviles permitiéndonos ganar tiempo consultándolo en cualquier momento, lugar o situación.

El uso de software libre además de garantizarnos estabilidad y seguridad nos libra del pago de licencias y nos asegura de cierto modo que tendremos acceso a actualizaciones y patchs de seguridad sin costo.

La metodología Scrum ha sido una parte importante en el desarrollo del sistema dado que se aplico dando soluciones sencillas a la aplicación de esta manera se podía ver en poco tiempo el sistema funcionando, simulando ser un prototipo o una maqueta a escala real (1:1) con el cual se podía manipular como si fuera el producto final de modo que era mas sencillo optimizar el código ya que las falencias eran visibles.

En el desarrollo de la aplicación, la motivación vino ha ser un efecto colateral a consecuencia del uso de Scrum. Al dividir la aplicación en módulos en cuestión de días la aplicación ya tomaba forma y se veía materializada la aplicación, motivando a seguir adelante y muchas veces hasta proyectando el tiempo restante dado los resultados actuales.

La planificación se facilito con Scrum, al desarrollar pequeños módulos semanales. Se podía proyectar el tiempo restante, dado que era posible evaluar la similitud de los módulos ya desarrollados con los que estaban por desarrollar

El uso de MVC, hace sencilla la actualización y el mantenimiento ya que nos aísla el requerimiento a manipular dejando el resto del código intacto, además del uso de codeigniter nos pone a nuestra disposición múltiples bibliotecas para hacer más fácil el desarrollo.

La interface gráfica ha sido diseñada pensando en la facilidad de uso de modo que el usuario se sienta familiarizado con la aplicación y casi no necesite el uso de manuales ya que el funcionamiento es similar en todas las instancias del programa haciéndolo intuitivo y útil.

Recomendaciones

El sistema usa un directorio de organizaciones muy complejo que lo relaciona con los proyectos ingresados es muy importante el respaldo frecuente de los datos ya que una pérdida de información ocasionaría un desperdicio de tiempo así como de días de trabajo.

Es importante la depuración de información ya que no evitara la perdida de tiempo relacionando datos obsoletos. Se propone una futura actualización en la que se puede registrar las consultas a determinada organización o proyecto de modo que se conozca si hay registros que nunca han sido usados o no son útiles para los proyectos ingresados.

Es posible añadir una métrica que nos indique una relación entre cuantos campos son requeridos por el proyecto y cuantos son los cubiertos por la organizaciones teniendo la calificación de 10/10 si un proyecto y una organización comparten el mismo numero de campos de interés.

Añadir una función, que me permita relacionar los proyectos, con los estudiante de la PUCE, es una actualización importante dado que se esta dejando fuera de la aplicación a esta eficaz colaboración. De esta manera se esta potencializando el sistema y la búsqueda estará enfocada en todos los proyectos y no solo en los que necesitan grandes recursos, además se tendría toda la información referente a los proyectos consolidada en una sola base de datos ocasionando una administración efectiva.

Al hacer actualizaciones o mantenimiento de la aplicación se recomienda usar el estándar usado en código para nombres de variables objetos y métodos, ademas de comentar el código cuando un método no sea tan claro.

Nunca están por demás los estándares de seguridad básicos como los cambios periódicos de claves de acceso y de los servidores, la validación de datos de entrada a fin de evitar la inyección de código malicioso entre otros.

Una buena retroalimentación entre los usuarios y el desarrollador referente a su funcionamiento pueden hacer de esta aplicación una verdadera herramienta de trabajo

Bibliografía

Software Libre . http://www.gnu.org/philosophy/free-sw.es.html Acceso: 27/12/2011

Software. http://es.wikipedia.org/wiki/Software#cite_note-4 . Acceso: 27/12/2011

Software. http://es.wikipedia.org/wiki/Software . Acceso: 27/12/2011

Apache. http://httpd.apache.org/. Acceso: 17/01/2012

php. http://www.php.net/manual/es/faq.general.php . Acceso: 4/01/2012

php. http://www.php.net/manual/es/preface.php . Acceso: 4/01/2012

MySql. http://www.mysql.com/why-mysql/ . Acceso: 11/01/2012

MySql. http://es.wikipedia.org/wiki/MySQL . Acceso: 11/01/2012

JavaScript. https://developer.mozilla.org/es/JavaScript/Acerca_de_JavaScript. Acceso: 13/01/2012

Programacion Extrema. http://www.programacionextrema.org/cgi-bin/wiki.pl?ProgramacionExtrema . Acceso: 17/01/2012

Programacion Extrema. http://es.wikipedia.org/wiki/Programación_extrema. Acceso: 17/01/2012

Mysql Query Browser Administrator. http://dev.mysql.com/doc/query-browser/es/mysql-query-browser-introduction.html. Acceso: 17/01/2012

Centro del Muchacho Trabajador. http://centromuchachotrabajador.org/wp/. Acceso: 25/01/2012.

Eazel Software Computer. http://en.wikipedia.org/wiki/Eazel Acceso 14/02/2012

Scrum.http://es.wikipedia.org/wiki/Archivo:EjemploDeDiagramaBurnDown.png Acceso 1/03/2012

Scrum. http://es.wikipedia.org/wiki/Scrum Acceso 1/03/2012

Andrew, Pham. SCRUM IN ACTION. Boston USA, CENGAGE Learning,2012

Elizabeth Woodward. A Practical Guide to Distributed Scrum,International Business Machines Cormporation,2010